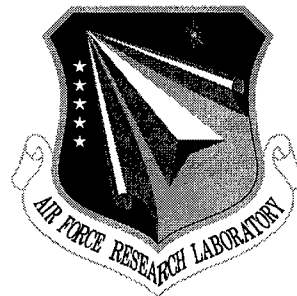


AFRL-IF-RS-TR-2000-148
Final Technical Report
October 2000



A DIALOGUE-BASED APPROACH TO MIXED-INITIATIVE PLAN MANAGEMENT

University of Rochester

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. C642

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

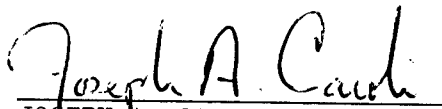
20010216 096

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK


This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-148 has been reviewed and is approved for publication.

APPROVED:


JOSEPH A. CAROLI
Project Engineer

FOR THE DIRECTOR:


NORTHROP FOWLER, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

A DIALOGUE-BASED APPROACH TO MIXED-INITIATIVE PLAN
MANAGEMENT

James F. Allen,
George M. Ferguson, and
Lenhart K. Schubert

Contractor: University of Rochester
Contract Number: F30602-95-1-0025
Effective Date of Contract: 01 June 1995
Contract Expiration Date: 01 July 2000
Short Title of Work: A Dialogue-Based Approach to
Mixed-Initiative Plan Management
Period of Work Covered: Jun 95 - Jul 00

Principal Investigator: James Allen
Phone: (716) 275-5288
AFRL Project Engineer: Joseph A. Caroli
Phone: (315) 330-4205

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION
UNLIMITED.

This research was supported by the Defense Advanced Research
Projects Agency of the Department of Defense and was monitored
by Joseph A. Caroli, AFRL/ITB, 525 Brooks Road, Rome, NY.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE OCTOBER 2000		3. REPORT TYPE AND DATES COVERED Final Jun 95 - Jul 00
4. TITLE AND SUBTITLE A DIALOGUE-BASED APPROACH TO MIXED-INITIATIVE PLAN MANAGEMENT			5. FUNDING NUMBERS C - F30602-95-1-0025 PE - 62301E and 63760E PR - C642 TA - 88 WU - 00	
6. AUTHOR(S) James F. Allen, George M. Ferguson, and Lenhart K. Schubert				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Rochester Computer Science Department 734 Computer Studies Building Rochester New York 14627-0226			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency Air Force Research Laboratory/IFTB 3701 N. Fairfax Drive 525 Brooks Road Arlington VA 22203-1714 Rome New York 13441-4505			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-148	
11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: Joseph A. Caroli/IFTB/(315) 330-4205				
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This research investigate ways to augment human plan reasoning capabilities in command and control tasks, such as logistics planning, crisis management and situation assessment. The goal was to avoid making the simplifying assumptions made in traditional research in planning that prevented effective human-computer collaborative planning. Specifically, no assumptions were made relative to goals being well specified in the beginning, or that the evaluation criteria can be precisely defined in a quantitative manner. The groundwork was laid for research in mixed-initiative planning through development of TRIPS (The Rochester Interactive Planning System), a robust speech-driven mixed initiative planning system, which has been installed at Global Infotech in Washington DC for the Command Post of the Future (CPOF) jumpstart demonstration as well as at the Air Force Research Lab in Rome, New York. TRIPS runs stand-alone without Rochester researchers present. New Theoretical techniques for speeding up well-founded domain-independent planners were also developed. In addition, simulation and data mining techniques to find improvements to large-scale plans that could not be reasoned about using traditional techniques were developed and demonstrated.				
14. SUBJECT TERMS Mixed-Initiative Planning, Dialogue, Human-Computer Interaction, Large-Scale Plans, Data Mining			15. NUMBER OF PAGES 84	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

1. Summary	1
2. Mixed Initiative Planning Systems	2
3. The Rochester Interactive Planning System (TRIPS)	4
4. Large Scale Planning	12
5. Improvements in Dialogue Modeling	16
6. Publications Describing Work Under this Grant	19

Appendices

A. Dialogue Systems: From Theory to Practice	23
B. An Architecture for a Generic Dialogue Shell	51

List of Figures

Figure 1:	TRIPS in operation	4
Figure 2:	The TRIPS Architecture	7

List of Tables

Table 1:	What happens in a mixed-initiative collaborative planning between humans	3
----------	--	---

1. Summary

In this project, we investigated ways to augment human plan reasoning capabilities in command and control tasks, such as logistics planning, crisis management and situation assessment. Our goal was to avoid making the simplifying assumptions made in traditional research in planning that prevented effective human-computer collaborative planning. Specifically, we didn't assume that the goals are well specified in the beginning, or that the evaluation criteria can be precisely defined in a quantitative manner. Rather, this information is developed incrementally through interaction with the human planner. With our focus on aiding humans in planning, we addressed problems such as how to present plans in an understandable manner and how to support extensive revision of plans as new goals and evaluation criteria become known. The human and machine are cooperating to build the plan, with each doing what they do best. We take this to be the key idea underlying mixed-initiative planning.

This project involved both theoretical work and system building. In addition to the original grant, we received supplemental funding for additional demonstration systems (the Command Post of the Future (CpoF) demo system) and for additional work on the dialogue aspects of the system. The main accomplishments of this project are described in this document. Specifically

- We laid the groundwork for research in dialogue-based mixed-initiative planning;
- We developed TRIPS, a robust speech-driven mixed-initiative planning system, which has been installed at Global Infotech in Washington for the CPOF jumpstart demonstration as well as at the Air Force Research Lab in Rome, NY. TRIPS runs standalone without any Rochester researchers present;
- We developed new theoretical techniques for speeding up well-founded domain-independent planners;
- We developed and demonstrated simulation and data mining techniques to find improvements to large-scale plans that could not be reasoned about using traditional techniques;

Under the supplemental funding for dialogue work,

- We developed new algorithms for pronoun resolution that outperform previous approaches on corpora-based tests;
- We developed a new architecture for real-time language generation in dialogue.

Each of these issues will be discussed briefly in the remainder of the report. More details can be found in the publications listed at the end.

2. Mixed-Initiative Planning Systems

One of the key results of this project was to better define the notion of mixed-initiative planning. The goal is to develop a system that can *enhance human performance* in managing plans. A decade ago, there was optimism that the long history of work in AI on planning could be harnessed to build effective planning tools. This hope disappeared, however, as it became clear that there was a serious mismatch between the way planning systems solve problems and the way humans solve problems. Automated planners required complete specifications of the goals and situation before starting to work, where people incrementally learned about the scenario and refined and modified their goals as the plan was being developed. Automated planners evaluated plans quantitatively and in black-and-white terms while humans used qualitative and subjective evaluations of plans. Automated planners worked on one solution at a time, whereas people compared options and alternatives before selecting a course of action.

Faced with this dilemma, we decided to try to design collaborative planning systems in which the user and machine collaborate to build plans, each providing the capabilities that it was best at. The human brought intuition, a notion of the goals and tradeoffs between goals, and highly developed problem solving strategies, while the machine brought an ability to manage detail, allocate and schedule resources, and perform quantitative analysis of proposed courses of action.

In order to determine how the system should interact with the human, we studied how humans interact with each other. We collected dialogues between two people interacting in a transportation planning scenario to see what types of interactions they used when they collaboratively planned. One person was given information about the domain, and the capabilities of vehicles, etc. The other was given goals to achieve and was ultimately responsible for the plan produced. The two could not see each other, and did not know each other. The only information they share at the start of the dialogue was an abstract map of a set of train routes. Table 1 summarizes the different classes of interaction and their frequency that we found in analyzing every utterance in one hour of a sample dialogue. Note that the only type of interactions supported by a typical state-of-the-art planning system (namely, adding a new course of action) represents less than 25% of the interactions. A good proportion of the interaction was concerned with maintaining the communication (e.g., summarizing, clarifying) or to manage the collaboration (e.g., discussing the problem solving strategy). Clearly, an effective collaborative planner would require much more than traditional planning technology.

Evaluating & comparing options	25%
Suggesting courses of action	23%
Clarifying and establishing state	13.5%
Clarifying or confirming the communication	13.5%
Discussing problem solving strategy	10%
Summarizing courses of action	8%
Identifying problems and alternatives	7%

Table 1: What happens in a mixed-initiative collaborative planning between humans

Faced with this analysis, we have focussed our research on several key areas. First, in order to provide the human with a convenient communication language, we have developed a spoken natural language dialogue interface. While a complex task in itself, it seemed more effective than trying to develop an expressive communication language that would then might require extensive user training. The success of the TRIPS systems validates these intuitions. Second, we developed more general models of plan reasoning and management, focusing on incremental development of plans, plan recognition, ways of managing and comparing different options, and ways of effectively communicating the structure and implications of proposed plans. Documentation on these efforts can be found at our website at www.cs.rochester/research/trains. An analysis of mixed-initiative interaction can be found Ferguson, Allen and Miller (1996) and in a recent *IEEE Intelligent Systems* article (Allen, 1999).

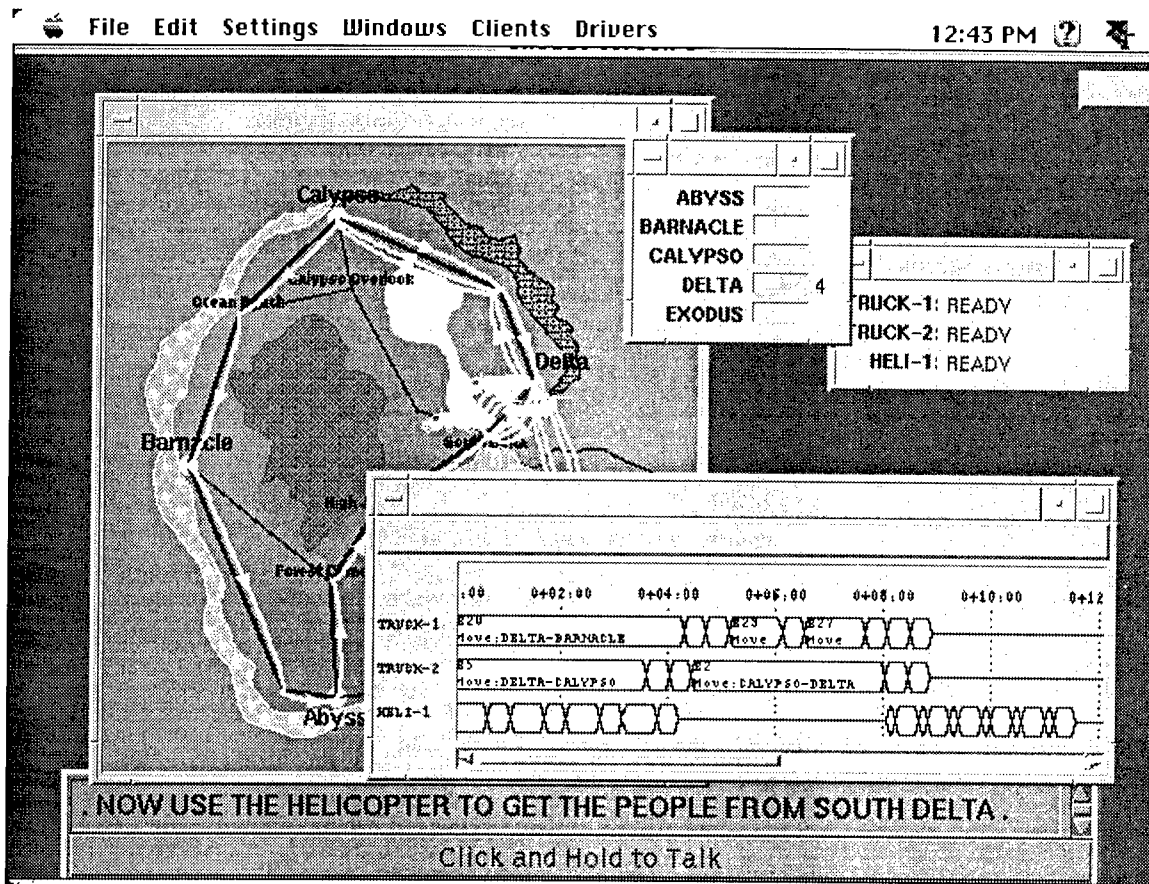


Figure 1: TRIPS in operation

3. The Rochester Interactive Planning System (TRIPS)

TRIPS, the Rochester Interactive Planning System, (Ferguson & Allen, 1998) is the culmination of our system building efforts under this grant. TRIPS supports dialogue-based mixed initiative planning in several different logistic planning domains. Our research plan involved designing and building a series of progressively more sophisticated systems working in progressively more realistic domains. TRIPS is the latest in a series of research prototypes. Its immediate predecessor, the TRAINS system, was one of the first systems to participate in an end-to-end conversation with a human to accomplish a planning-like task. Compared to TRAINS:

- TRIPS operates in a significantly more complex logistics and transportation world, with cargos being delivered using a variety of vehicles.
- TRIPS performs significantly more complex planning, including coordinating the actions of multiple agents, reasoning about temporally extended actions, handling some resource constraints, and so on, all as quickly as possible.

- TRIPS has a more sophisticated graphical view of the plan(s) under consideration, including a construction window view (timeline) modeled after Adobe Premiere's and task palettes modeled after Photoshop's layers. It also has a few types of charts that can be used to answer certain kinds of queries from the manager.
- TRIPS supports "what-if" scenarios for comparing options, for example "What if we went along the coast?" or "What if we used a helicopter instead?"
- TRIPS includes a simulator module that performs multi-run stochastic simulation of plans, gathers statistics on action executions and the world state, and produces results that can be viewed with the same tools. It also performs data-mining to extract events that significantly affect plan success or failure.

TRIPS, like TRAINS, is a multi-process, fully-networkable set of modules that exchange Knowledge Query and Manipulation Language (KQML) messages through a general-purpose Input Manager. In TRIPS, even more of the functionality is divided into separate modules, improving modularity, development and debugging, and often performance.

TRIPS is an assistant to a human manager. The two of them collaborate to construct plans in crisis situations. One example scenario involves the island of Pacifica (see Figure 1) which is in the path of an approaching hurricane. Because of the hurricane, the population must be evacuated to one of the cities (the safe city) and from there presumably to safety. The manager's task is to plan the evacuation, using a variety of vehicles (with varying capabilities) at his or her disposal. There may be a variety of constraints placed on the final plans, such as time, cost, weather-related factors, and so on. The important features of this scenario as a research environment are: (a) it is clear what the task is and when it is completed (i.e., we can easily describe tasks to people, they can know when they're done, and we can verify that); (b) the relative quality of the solution is easily measured (in terms of time, violated constraints, or some measure of robustness, for example); and (c) it is easy to vary the complexity of the task from trivially solvable to problems that people would have difficulty solving and, in fact, that TRIPS could not solve on its own (e.g., increase numbers of people, change numbers and types of vehicles, modify road network, and so on). The first two aspects make evaluation possible, the final aspect lets us work incrementally on progressively harder problems.

Example Dialogue

An example dialogue is presented below. This is a verbatim transcript of an actual session, annotated to describe the graphical displays generated by TRIPS. The times shown are elapsed time from the beginning of the dialogue. Note that most utterances are understood within a few seconds, including speech recognition and all processing. This level of responsiveness is required to maintain an effective dialogue with the user.

0:00 H: Hello.

0:02 S: Hi there.

0:03 H: Show me a map of Pacifica.

0:07 S: Ok. *Displays large map of the island of Pacifica*

0:08 H: Where are the people?

0:11 S: There are two groups of people at Exodus, two at Calypso, two at Barnacle, and one group of people at Abyss. *Displays a bar chart showing the numbers of people in the cities. This chart is updated as the conversation progresses to show the state of the world given the plan under consideration.*

0:17 H: And where are the transports?

0:19 S: There are two trucks and one helicopter at Delta.
Displays a chart showing vehicle status and contents (also up-dated through the conversation)

0:24 H: Use a truck to get the people from Calypso to Delta.

0:31 S: Your wish is my command. *Displays a construction window with a timeline view of the plan under development with the vehicles on the vertical axis and time horizontally. The map is also updated to show the route taken to Calypso and back.*

0:34 H: How long will that take?

0:37 S: It will take six hours and forty-two minutes.

0:43 H: What if we went along the coast instead?

0:47 S: That option would take ten hours and twenty-seven minutes.
Displays another construction window showing the option alongside the original plan.

0:53 H: Forget it.

0:55 S: Ok. *Deletes second construction window from the display.*

1:05 H: Use the other truck to get the people from Exodus to Delta.

1:13 S: Ok. *Adds the new task to the original construction window, using a different color, and updates the map. In this case, the task is clearly seen to take more than ten hours.*

1:22 H: Let's use the helicopter instead.

1:34 S: Ok. *Changes the construction window view of this task and updates the map display.*

1:38 H: Now send truck one to Barnacle.

1:41 S: Your wish is my command.
Adds a third task to the construction window (in a third color) and updates the map.

1:44 H: Pick up the people there.

1:48 S: Ok. *Updates displays*

1:50 H: Go on to Abyss.

1:54 S: Ok. *Updates displays*

1:57 H: And take the people there to Delta.

2:05 S: Ok. *Updates displays*

2:10 H: That looks good.

2:12 S: Glad to be of service.
We now suppose that the manager learns of a change in road status from an external source, and informs the system so that the plan can be adjusted. Delta Bridge crosses the inlet between Delta and South Delta.

2:18 H: Delta bridge is out.

2:21 S: Ok. The highlighted actions may need to be revised.
Flashes Delta Bridge in red, and highlights actions whose routes use the bridge in red in the construction window.

2:37 H: Send truck two along the coast instead.

2:39 S: Ok. *Displays new route (to Calypso and back) and updates the construction window (longer trips).*

2:42 H: Have the other truck stop at South Delta instead.

3:02 S: Ok. *Display new route (along coast to Barnacle, then to Abyss, then to South Delta) and updates the construction window.*

3:06 H: Unload the people there.

3:13 S: Ok. *Updates displays*

3:18 H: And use the helicopter to get the people from South Delta to Delta.

4:03 S: No problem. *Multiple flights with the helicopter are added to the third task in the construction window and the map is updated to show them.*

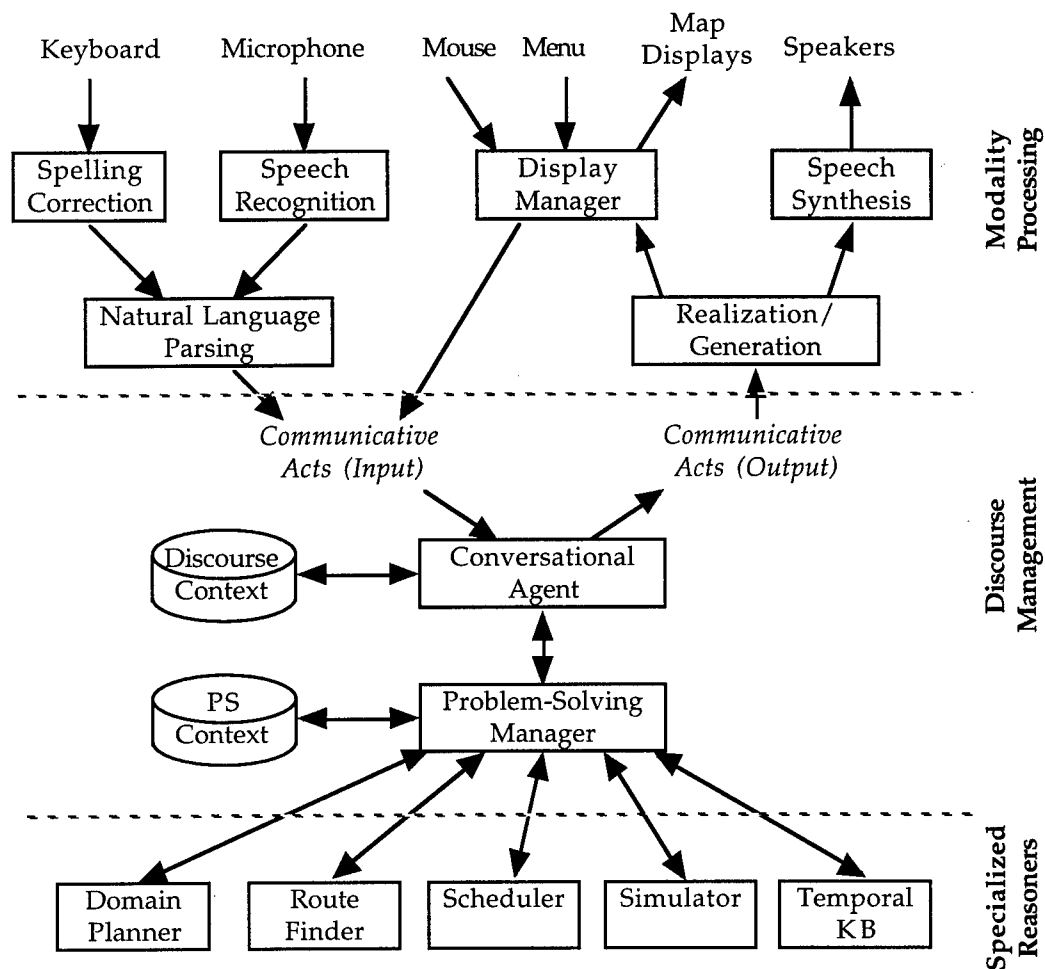


Figure 2: The TRIPS Architecture

TRIPS System Architecture

The organization of TRIPS is shown in Figure 2. Modules communicate by exchanging KQML messages using our own central message-passing Input Manager (not shown in the figure). Most modules are in fact separate Unix processes, and TRIPS can run on any combination of machines that can run the individual modules. The TRIPS infrastructure allows any program that can read standard input and write standard output to exchange messages.

As shown in the figure, the components of TRIPS can be divided into three groups:

- **Modality Processing:** This includes speech recognition and generation, graphical displays and gestures, typed input, and so on. All modalities are treated uniformly. For input, words and gestures are parsed into meaning representations based on

treating them as communicative acts. For output, communicative acts generated by the system are realized using speech or graphics.

- **Dialogue Management:** These components are the core of TRIPS, and are responsible for managing the ongoing conversation, interpreting user communication in context, requesting and coordinating specialized reasoners to address the needs of the conversation, and selecting what communicative actions to perform in response.
- **Specialized Reasoners:** These components provide the brains of TRIPS, in the sense of being able to solve hard problems such as planning courses of actions, scheduling sets of events, or simulating the execution of plans. The goal here is to provide a form of plug-and-play interoperability, where new or improved specialized reasoners (including, for example, network-based sources or agents) can be easily added to the suite of resources at TRIPS disposal.

Interaction as Conversation

The Conversational Agent (CA) coordinates all system activity as it interacts with the user. The key idea for integrating various different input and output modalities is that all user interaction is viewed as *communicative acts*, a generalization of speech acts. As a consequence, all communication between the Conversational Agent and the modality processing modules is in terms of possible communicative acts that have been or should be performed. TRIPS supports a wide range of speech acts, ranging from direct requests (e.g., *show me the map*), questions (*where are the transports?*), assertions (*The bridge is out*), suggestions (*Lets use a helicopter instead*), acceptances and rejections (*ok, no*), as well as a range of social acts including thanks, apologies, and greetings. In addition, there are a limited range of gestural acts such as pointing and dragging screen objects using the mouse. The modality processing modules typically produce a set of possible *surface acts* based on the form of the act. The Conversational Agent then combines these interpretations with the discourse context in order to determine the *intended acts*, and then in coordination with the problem solving manager, determines the systems response, which again is expressed in terms of communicative acts.

Since the task in TRIPS is interactive problem solving, most of the acts that are performed relate to different problem solving operations. The operations most common in TRIPS include introducing a new task or subtask to the plan, modifying or deleting an existing task, defining all or part of a solution for a task, modifying an existing solution, or evaluating all or part of the plan. As well, as is common in human-human problem-solving, either the person or the system can create alternate solutions (options) for comparison purposes, remove an option from consideration, or adopt a particular option. Note that the problem solving operation involved is orthogonal to the communicative act

used to communicate it. For example, one might request a modification to a plan, suggest a modification, accept or reject a modification, or promise to do a modification.

The Conversational Agent is driven by a set of rules that identify possible interpretations intended by the user and plan a system response for each. These interpretation/response pairs are then ranked and the system's response is selected. Currently, this selection process is based on a static ranking of the strength and reliability of the rules. In the future, more complex deliberation processes will be introduced that allow the system to generate a wider range of responses, including taking the initiative in the conversation if desired.

Integration as Collaborative Problem-Solving

While the Conversational Agent coordinates the interpretation of communicative acts and chooses system responses, it does not have direct knowledge of the task or current state of the problem solving process. This information is handled by the Problem Solving Manager, which maintains an abstract representation of the task and the current solution (or solutions) under consideration, and coordinates reasoning by the specialized reasoners when necessary. The key idea supporting the PSM in this is a very general representation of plan-related information, including a hierarchical task structure, explicit representation of the possible solutions under consideration, and a temporal knowledge base that represents the world over time relative to different solutions.

The abstract plan representation is general purpose and is used for a wide range of purposes including: providing the context for recognition of the users intentions, driving displays that summarize the state of the world and/or plan, answering queries about the plan, building the commands and context required by the specialized reasoners, and integrating and managing the results from all the specialized reasoners. Note that while it supports a wide range of purposes, this representation is not typically used directly in any planning process. Rather the PSM converts the general representation into specific commands to each specialized reasoner using a representation that that reasoner understands. It then converts the results returned back into the general representation for use by other components. A key feature of this representation is that it allows the system to represent and reason about plans that it could not have generated on its own. This allows the user to incrementally develop plans that are more complex than the TRIPS planner can generate.

To support the interpretation processes in the conversational agent, the PSM and conversational agent interact using a propose-evaluate-confirm protocol. The conversational agent suggests a possible problem solving action, say to modify a certain subtask by using a different vehicle. The PSM then evaluates this interpretation based on

coherence (i.e., did this operation make sense in the current problem solving context) and *feasibility* (i.e., could the requested operation be performed). Based on these evaluations, the CA chooses a particular set of interpretations and then informs the PSM so it can update its context for the next interaction. In order to evaluate each interpretation, the PSM may invoke the specialized reasoners as necessary to perform the actual reasoning required.

Consider a brief example from the sample dialogue, when the user says *Lets use the helicopter instead*. From its surface form, this utterance is interpreted as a suggestion to use the helicopter in some unspecified part of the plan instead of some other (also unspecified) vehicle. The CA uses coherence heuristics to explore the most likely possibility: that the modification should apply to the last subtask being discussed, namely getting the people from Exodus to Delta. It asks the PSM to evaluate modifying this plan by replacing something with the helicopter. The PSM uses its abstract representation of the plan to find likely objects that the helicopter could replace in this task, in this case the truck mentioned in the previous interaction. It then calls the Planner with the task to replace this truck with the helicopter in this task. The Planner performs this operation and returns a revised plan. This plan, however, is quite different from the previous solution. Rather than making one trip with the truck, the plan now involves making two trips with the helicopter (due to different vehicle capacities). The PSM then calls the Router to instantiate routes for the helicopter for its two trips, and then invokes the Scheduler to produce a nominal plan that can be used to generate a display. Since all these operations were performed successfully, the PSM returns to the CA an evaluation saying that this interpretation ranks high on both coherence and feasibility. The CA has no other viable options, and so notifies the PSM that this is the interpretation selected. The PSM updates its problem solving state to reflect the new plan, and updates the world model used to drive the plan display. The CA executes the response corresponding to this interpretation, causing display updates and a spoken confirmation.

Installations and Portability Issues

As the culmination of several years of work in system building, we delivered a version of TRIPS to Global Infotech in Washington DC and the Air Force Research Lab in Rome New York. This is a complete end-to-end spoken dialogue system that can be run and demonstrated with only minimal training. The version at GITI has been demonstrated extensively as part of the Command Post of the Future programs jumpstart.

In order to allow TRIPS to be demonstrated in many different domains, the system is driven off files that define the scenario for the demo, including the maps, key locations, objects involved, planning operators that can be used in problem solving, and additions to the lexicon (mostly names of all the new objects in the domain). By loading a different

scenario file, TRIPS can solve different problems in different situations. TRIPS currently runs in three different domains, although the robustness in each domain varies. These are

Pacifica - the original domain for the TRIPS system involving the evacuation of an island in the face of an incoming hurricane. TRIPS is quite robust in this domain and it supports novices solving novel problems;

Command Post of the Future - this domain involves planning the deployment of engineering forces to clear an airfield in a battle situation. This demo is quite closely scripted to fit in with the overall CPOF demonstration.

Airlift - this domain involves planning airlifts and involves integrating TRIPS with realistic airlift planning tools developed at BBN (the CAMPS system). This effort is primarily being supported by the CoABS program, although the language extensions have been funded by the current effort (Burstien, Ferguson and Allen, 2000).

Conclusions

Integrated systems like TRIPS take time and effort to build. What have we gained from the experience?

First, we can evaluate performance by having people solve problems. In work on TRIPS' predecessor TRAINS, we developed a methodology for testing groups of naive users solving a set of predefined problems (Sikorski & Allen, 1995, Stent & Allen, 1996). The results were encouraging (90% of sessions resulting in success), but these were mostly a reflection of the simplicity of the TRAINS task. We have not yet evaluated TRIPS on its more complex task and domain, but applying the same methodology, we expect to be able to show that people can solve problems faster with TRIPS than with another person. We hope to report those results in the near future. The point for this report, however, is that this experiment could not even be conceived without an integrated, end-to-end system with which to work.

Second, the emphasis on integrating multiple specialized reasoners at the problem-solving level has had several benefits. The drive towards integration has resulted in a very general shared representation of plans, objectives, domain objects, and the like. This representation is used by a range of components from natural language understanding to planning and simulation. And of course, the integration of multiple specialized reasoners allows us to plug new technologies into TRIPS or interact with external agents. The key to making this more generally effective is the specification of component or agent *capabilities* so that meta-reasoners like the TRIPS PSM can task them and understand their responses. Specifying and using such capabilities is an active area of our current research.

Finally, the close, intuitive integration between person and computer in TRIPS has several benefits. Viewing the interaction as a conversation is far more natural for the person than learning arcane command languages or GUIs. This will translate into more effective performance with less training (as we hope to show in our evaluation experiments). The closely-integrated, mixed-initiative interaction is also easier for the computer, since it is possible for it to indicate when it can't solve a problem and literally ask for help. The generality of the representations described above ensures that the system can understand and use suggestions that it would never have come up with by itself. The emphasis for the specialized reasoners then changes from complete but impractical reasoning, to flexible and expressive but almost certainly incomplete forms of reasoning.

In the end, the whole, integrated system is greater than the sum of its parts. TRIPS allows the human and the system to collaboratively solve harder problems than either could solve on their own.

4. Large-Scale Planning

The plans that are used in logistic domains are several orders of magnitude greater than the plans that can be constructed using traditional planning techniques. We explored a number of different approaches to addressing this problem. We developed new techniques based on simulation and data-mining that allowed us to evaluate and improve large plans, and we developed new techniques for speeding up traditional planning algorithms.

4.1 Simulation and Data Mining

We developed new plan-reasoning techniques involving simulation and data mining to incrementally improve large plans, as described in Lesh and Allen (1998, 1999). Past research on assessing and improving plans in domains that contain uncertainty has focused on analytic techniques that are exponential in the length of the plan. Little work has been done on choosing from among the many ways in which a plan can be improved. We developed the **Improve** algorithm that simulates the execution of large, probabilistic plans. **Improve** runs a data mining algorithm, described below, on the execution traces to pinpoint defects in the plan that most often lead to plan failure. Finally, **Improve** applies qualitative reasoning and plan adaptation algorithms to modify the plan to correct these defects. We tested **Improve** on plans containing over 250 steps in an evacuation domain, produced by a domain-specific scheduling routine. In these experiments, the modified

plans had over a 15% higher probability of achieving their goal than the original plan (based on simulations of the plan executions).

The **Improve** algorithm is appropriate for domains in which there are probabilistic (or uncertain or stochastic) effects of actions or exogenous events. In order to apply **Improve**, we need a fairly accurate model of the domain and a method for generating a reasonable, but sub-optimal plans. For example, **Improve** might be appropriate for a large manufacturing domain for which there currently exists an adequate schedule or plan but the plan can be improved by re-arranging the schedule or adding maintenance actions.

We have applied sequence mining techniques to develop an algorithm called **PlanMine** that extracts patterns of events that predict failures from databases of plan executions. The plan executions can be either real or simulated. New techniques were needed to solve this problem because previous data mining algorithms were overwhelmed by the staggering number of very frequent, but entirely unpredictable patterns that exist in the highly structured database of plan executions. We have developed techniques that reduce the size of the returned rule set by three orders of magnitude, in our experiments.

The **PlanMine** algorithm is appropriate for extracting patterns that predict failure for any process that we can simulate repeatedly and accurately. For example, suppose we had a traffic simulator that could produce sample traffic patterns. We could use PlanMine to detect patterns of traffic that predict that a traffic jam will occur in the near future. These patterns can be used either to help a person understand the domain better or to build predictive monitors or alarms that signal when a traffic jam is likely to occur, before it occurs. In the planning domain, we were able to build monitors that could accurately predict plan failures before they occurred.

More details on this work can be found in Lesh and Allen (1998), Lesh, Martin and Allen (1998), Lesh and Allen (1999) and Lesh, Zaki and Ogihara (1999).

4.2 Improving Search in Planning

We undertook a series of theoretical and experimental studies to bring domain-independent planners closer to practicality. The techniques studied include the use of improved techniques for A*-like search of the plan space, the use of precomputed sets of possible parameter values of the operators available for achieving goals, and most recently, the automated discovery of domain constraints (state invariants) from operator structure and the initial state, and use of these constraints to reduce search in planning. Each of the techniques studied was shown to provide speedups of several orders of magnitude for many standard test problems. The automated discovery of state constraints, implemented in a package called DISCOPLAN, also holds promise for domain knowledge engineering, since it allows a domain designer to rapidly discover and

examine the domain properties that are implicit in a proposed set of actions (operators) in a domain, and a given starting state.

This research, outlined under the next two subheadings, has led to 3 conference papers and a journal paper within the time frame of the grant, and two more papers (based largely on work done up till 1999) appear in year 2000 conferences.

Improving Search and Using Pre-computed Sets of Parameter Values

We formulated, implemented and tested several techniques that can dramatically improve the performance of domain-independent partial-order planners such as UCPOP. One technique combines an improved A* search of the plan space with a “zero commitment” goal selection strategy. The strategy uses S+OC—the number of steps in a plan plus the number of open conditions still to be established—as a heuristic measure for UCPOP's A*-search of the plan space. The flaw-selection strategy, which is termed ZLIFO, prefers zero commitment plan refinements to others, and otherwise uses a LIFO (stack) discipline. Zero commitment refinements are logically necessary ones: they either eliminate a plan altogether because it contains an irremediable flaw, or they add a unique step or unique causal link (from the initial state) to establish an open condition that cannot be established in any other way. Experimental tests were performed with slightly modified versions of UCPOP (because of that system's ready availability—but the techniques could be used with many other planners). For the more difficult problems taken from the available UCPOP test suite and elsewhere, improvements by factors ranging from 5 to more than 1000 were obtained, with the hardest problems giving the greatest improvements (Schubert & Gerevini 1995, Gerevini & Schubert, 1996a).

A more radical technique is the use of operator parameter domains to prune search. These domains are initially computed from the definitions of the operators and the initial and goal conditions, using a polynomial-time algorithm that propagates sets of constants through the operator graph, starting in the initial conditions. During planning, parameter domains can be used to prune nonviable operator instances and to remove spurious clobbering threats. In further experiments based on modifications of UCPOP, this pruning technique often gave speedups by an order of magnitude or more for difficult problems, both with the default UCPOP search strategy and with the improved strategy (Gerevini & Schubert 1996b).

Automatically Discovering Domain Constraints

The effectiveness of using precomputed parameter domains led us to investigate more general preprocessing techniques that discover properties of planning domains and thereby enable greatly improved planing, as well as providing guidance in problem domain engineering. We developed a set of techniques aimed at discovering state

constraints from the structure of planning operators and the initial state. These included (a) "implicative constraints" (for example, an inferable implicative constraint in some blocks world problems is that whenever an object *x* is ON an object *y*, then *y* is NOT CLEAR, unless *y* is the TABLE); and (b) simultaneous implicative and single-valuedness (sv-) constraints (such as the constraint in the TRAINS transportation planning world that if cargo *x* is IN car *y*, then *y* is NOT EMPTY, and furthermore, *x* is IN no other car).

The discovery techniques consist of generating hypotheses by inspection of operator effects and preconditions, and checking each hypothesis against all operators and the initial conditions. In addition, Schubert and Gerevini developed a method of inferring predicate domains by forward propagation of ground literals from the initial state, in Graphplan-like manner. The various techniques are implemented in a package called DISCOPLAN. The potential of these techniques for improved planning was demonstrated by adding computed state constraints and predicate domains to the specification of problems for SAT-based planners. The results showed that large speedups in planning can be obtained by such automated methods, potentially obviating the need for adding hand-coded state constraints (Gerevini & Schubert 1998).

This work was subsequently extended in both practical and theoretical directions. An across-the-board extension was the allowance for conditional effects in operators specifying a planning domain. This is likely to be of considerable practical interest in domain-independent planning, since formalisms permitting operators with conditional effects (e.g., UCPOP and PDDL) facilitate compact encoding of complex operators that would otherwise require unnatural and potentially very large expansions as multiple unconditional operators. The algorithms and implementation for (a) and (b) above were generalized accordingly, and the newly implemented techniques similarly allow for conditional effects. The new practical capabilities of DISCOPLAN include the discovery of (1) subtype and mutual exclusion constraints among static monadic predicates; (2) single-valuedness (sv-) constraints derivable in isolation; (3) implicative constraints, simultaneously with sv-constraints, where the antecedent and consequent may each contain variables not contained in the other (and all variables are universally quantified); and (4) antisymmetry constraints. Furthermore, DISCOPLAN is able to infer additional constraints of the type just mentioned by "expanding" operators so as to include preconditions and effects implied by constraints discovered earlier, and then re-running the discovery algorithms. All of the types of constraints discovered by DISCOPLAN are known (from other studies in the literature, as well the above 1998 work of Gerevini & Schubert) to be very useful in speeding up planning and giving insight into the structure of a task domain. (see Gerevini & Schubert 2000a,b).

5. Improvements in Dialogue Modeling

We received supplemental funding in the last year of this contract to continue work on the dialogue modeling aspects of the TRIPS system. Here we describe two areas of progress: developing new reference resolution algorithms that have better coverage of dialogue than any previous approaches, and laying the groundwork for real-time generation in dialogue settings.

Reference Resolution

While many pronoun interpretation algorithms have been developed over the last twenty years, they have tended to cover a very narrow range of phenomena and have not been rigorously tested. Our method has involved analyzing naturally occurring human/human discourse in order to shed light on the principles underlying pronoun use by humans, and to create programs that implement these principles.

Understanding pronouns is crucial for any natural language understanding task. It is especially important in dialog, which has a higher density of pronouns than does written language. Many pronoun interpretation algorithms already exist; however, they tend to cover a very narrow range of phenomena, tend not to be informed by current linguistic theory, and tend to be formulated for written monologue rather than dialogue. Our method involves analyzing naturally occurring discourse in order to shed light on the linguistic principles underlying pronoun use by humans, to create programs that implement these principles, and to evaluate existing algorithms to isolate their deficiencies when faced with spoken language.

Previous models for pronoun interpretation had the following limitations:

- They only resolve definite (such as he/it), not demonstrative (such as this/those) pronouns, and only the definite pronouns whose antecedent is a noun phrase. This approach ignores an entire class of pronouns that refer to stretches of text, propositions, events, etc.
- They rely on calculations of salience of noun-phrase-mentioned entities, ignoring semantic information.
- They rely on full syntactic analysis and include notions of sentence boundaries in their definitions. Both assumptions are inappropriate for processing spoken dialogue.

Our algorithms for pronoun resolution fall into two main categories. The first thrust concerns definite pronouns (i.e. it/he), which can be modeled as a search for the most salient, or focused item. The LRC centering algorithm (Tetreault, 1999) processes incrementally (as it gets to each pronoun) and therefore does not require a complete sentence as input. This makes it more appropriate for processing spoken input compared to previous centering-based pronoun resolution algorithms. Our attempt to apply

centering to spoken language showed that the theory did not adequately account for spoken dialogue phenomena. Our tests show that LRC performs better than four leading pronoun resolution methods.

The second thrust of this research is to create a model that expands the range of pronominal phenomena for which an interpretation can be computed. No previous computational model exists that can account for pronominal reference to 'abstract entities' such as the proposition expressed by a sentence, the speech act of the sentence, items from the problem-solving state, etc. To interpret these pronouns requires a completely new approach because they do not necessarily have antecedents in the discourse. After analyzing these pronouns in the TRAINS93 corpus (see Byron and Allen, 1998), we designed a first-pass algorithm that can interpret these pronouns. The algorithm utilizes findings from theoretical linguistics as well as observations from data analysis on several discourse genres, and relies on calculating the semantic type of the referent for the pronoun. The model has not been implemented yet, but we are in the process of implementing it within the TRIPS system. A hand-simulation of the algorithm correctly resolved approximately twice as many pronouns compared to a baseline method.

For much of this work, adequate test data to use in evaluating our programs was not available. Therefore, this year we had a team working on data collection to create a set of texts with the correct antecedents of pronouns indicated. These texts will be used to score how well our programs for pronoun interpretation work. The process of this data collection is very good for the analysts involved - increasing their exposure to naturally occurring linguistic phenomena and improving their ability to build hypotheses for how to process that data.

The following were our major accomplishments in this area:

- We have created a large testbed system (Byron and Tetreault, 1999) for testing pronoun resolution methods. The testbed can be configured at runtime to accept input from a variety of sources and to resolve pronouns with a variety of pronoun resolution algorithms. As part of that effort, we created the following programs:
- Coded the "left-right centering" algorithm for pronoun resolution, detailed in the ACL98 paper.
- We coded around 20 independent modules to calculate salience based on different discourse properties, such as being the subject of the previous sentence.
- Coded a genetic algorithm program that seeks to determine an optimal combination of salience factors of individual items mentioned in the text. This work is discussed in (Byron and Allen, 1999).

Real-time Dialogue Generation

One weakness of TRIPS has been on the natural language generation side, specifically in planning the content for multimodal, mixed-initiative dialogue. We have been focusing on developing techniques for managing useful subdialogs, and successful shifts in initiative. The work so far has been pragmatic and empirical. We have succeeded in producing natural-sounding short clarification subdialogues, but this has been done using heuristic methods. We completed a study of the literature and identified several difficulties with simply extending content-planning theories for text to dialogue (see Stent 1999).

We completed the collection of a corpus of 20 human-human dialogs in which the people worked to allocate resources for emergency services in Monroe County. This data is now in the process of being transcribed and annotated. These dialogs are being studied to give indications of what types of communication need to be planned for by a generation module, and how they can be planned. In the future, this corpus will be made available to the larger research community. Our initial results were discussed at a workshop (Stent 1999b) in Germany in 1999.

6. Publications Describing Work Under This Grant

- Allen, J.F., Bradford W. Miller, Eric K. Ringger, and Teresa Sikorski. "Robust Understanding in a Dialogue System," *Proc. Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 62-70, 25-27 June, 1996.
- Allen, J.F. and George Ferguson, "Events and Actions in Interval Temporal Logic", in O. Stock (ed.), *Space and Time*, Springer-Verlag, 1987.
- Allen, J.F., "Natural Language Technology", *McGraw-Hill Year Book on Science and Technology*, McGraw-Hill, to appear 1998
- Allen, J.F. et al, "TRAINS: Towards Practical Dialogue Systems", *Handbook of Natural Language Processing*, Herman Moisl, Harry Somers and Robert Dale (eds), Marcel Dekker Inc, 1998.
- Allen, J.F., "AI Growing Up," *AI Magazine* 19, 4, Winter 1998.
- Allen, J.F., G.M. Ferguson, B.W. Miller, E.K. Ringger, and T.M. Sikorski (Zollo), "TRAINS: Towards practical dialogue aystems," in R. Dale, H. Moisl, and H. Somers (Eds.). *A Handbook of Natural Language Processing*. Marcel Dekker, Inc., 1998.
- Allen, J.F. "Mixed-Initiative Interaction, Trends and Controversies." *IEEE Intelligent Systems* 14(5), (1999)
- Burstein, Mark, George Ferguson, and James Allen, Integrating Agent-Based Mixed-Initiative Control With An Existing Multi-Agent Planning System, ICMAS-2000.
- Byron, D.K. and J.F. Allen, "Applying genetic algorithms to pronoun resolution," *Proc., 16th Nat'l. Conf. on Artificial Intelligence (AAAI-99)*, Orlando, FL, July 1999.
- Byron, D.K. and J.F. Allen, "Resolving demonstrative anaphora in the TRAINS 93 corpus," *Proc., DAARRC2—Discourse, Anaphora and Reference Resolution Colloquium 2*, Lancaster U., August 1998.
- Byron, D.K. and Peter Heeman, "Discourse Markers in Task-Oriented Spoken Dialog" in *Proceedings of the Fifth Biennial European Conference on Speech Communication Technology (Eurospeech '97)*, September 1997.
- Byron, D.K. and J.R. Tetreault, "A flexible architecture for reference resolution," *Proc., 9th Conf. of the European Chapter of the Assoc'n. for Computational Linguistics (EACL-99)*, Bergen, Norway, June 1999.
- Core, M.G. and L.K. Schubert, "Speech repairs: A parsing perspective," *ICPhS Satellite Meeting on Disfluency in Spontaneous Speech*, U. California, Berkeley, July 1999.

- Core, M.G. and L.K. Schubert, "A syntactic framework for speech repairs and other disruptions," *Proc., 37th Annual Meeting of the Assoc'n. for Computational Linguistics (ACL-99)*, College Park, MD, June 1999.
- Ferguson, G.M. and J.F. Allen, "TRIPS: An integrated intelligent problem-solving assistant," *Proc., 15th Nat'l. Conf. on Artificial Intelligence (AAAI-98)*, Madison, WI, July 1998.
- Ferguson, G., James Allen, and Brad Miller, "TRAINS-95: Towards a Mixed-Initiative Planning Assistant," *Proc. Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70-77, Edinburgh, Scotland, 29-31 May, 1996.
- Ferguson, G., J.F. Allen, B.W. Miller, E.K. Ringger, The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant, TRAINS Technical Note 96-5, Dept. of Computer Science, University of Rochester.
- Galescu, L. Eric Ringger and James Allen. "Rapid Language Development for New Task Domains", *Proc. ELRA First intl Conf. on Language resources and Evaluation*, Granada, Spain, May 1998.
- Galescu, L. and E.K. Ringger, "Augmenting words with linguistic information for n-gram language models," *Proc., 6th European Conf. on Speech Communication and Technology (Eurospeech-99)*, Budapest, September 1999.
- Gerevini, A. and L.K. Schubert (2000b), "Discovering State Constraints in DISCOPLAN: Some New Results", *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI'2000)*, July 30-August 3, Austin, Texas.
- Gerevini, A. and L.K. Schubert (2000a), "Extending the types of state constraints discovered by DISCOPLAN", *Proc. of the AIPS 2000 Workshop on Analyzing and Exploiting Domain Knowledge for Efficient Planning*, Breckenridge, Colorado, USA, April 14-17, 2000.
- Gerevini, A. and L.K. Schubert (1998), "Inferring state constraints for domain-independent planning", *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*, July 26-30, Madison, WI.
- Gerevini, A. and L.K. Schubert (1996b), "Accelerating partial-order planners: Some techniques for effective search control and pruning", *J. of Artificial Intelligence Research* 5, pp. 95-137, Sept. 1996.
- Gerevini, A. and L.K. Schubert (1996a), "Computing parameter domains as an aid to planning", *Proc. of the 3rd Int. Conf. on Artif. Intell. Planning Systems (AIPS-96)* (B. Drabble, ed.), May 29-31, Edinburgh, The AAAI Press, Menlo Park, CA, pp. 94-101.
- Heeman, P.A. and J.F. Allen, "Speech repairs, intonational phrases and discourse

- markers: Modeling speaker's utterances in spoken dialog", *Computational Linguistics*, 1999.
- Kaplan, A.N. and L.K. Schubert, "Simulative inference in a computational model of belief," to appear in H. Bunt and R. Muskens (Eds.). *Computational Semantics. Studies in Linguistics and Philosophy Series*. Kluwer Academic Publishers.
- Lesh, N. and J.F. Allen, "Simulation-based inference for plan monitoring," *Proc., 16th Nat'l. Conf. on Artificial Intelligence (AAAI-99)*, Orlando, FL, July 1999.
- Lesh, N., N.G. Martin, and J.F. Allen, "Improving big plans," *Proc., 15th Nat'l. Conf. on Artificial Intelligence (AAAI '98)*, 860-867, Madison, WI, July-August 1998.
- Lesh, N. and James Allen, Simulation-based inference for plan monitoring, *Proc. National Conference on AI, (AAAI-99)*, Orlando, FL, 1999.
- Lesh, N., M.J. Zaki, and M. Ogihara, "Mining features for sequence classification," *5th ACM SIGKDD Int'l. Conf. on Knowledge Discovery and Data Mining (KDD)*, San Diego, CA, August 1999.
- Schubert, L.K., "Dynamic skolemization," in H. Bunt and R. Muskens (Eds.). *Computational Semantics. Studies in Linguistics and Philosophy Series*. Kluwer Academic Publishers.
- Schubert, L.K. and A. Gerevini (1995), "Accelerating partial order planners by improving plan and goal choices", *Proc. of the 7th Int. Conf. on Tools with AI (ICTAI'95)*, Nov. 5-8, Herndon, VA (Hyatt Dulles Hotel), 1995, pp. 442-450.
- Seagull, A.B. and L.K. Schubert, "Guiding a well-founded parser with corpus statistics," *Proc., Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora* (held at *37th Annual Meeting of the Assoc'n. for Computational Linguistics (ACL-99)*), 179-186, College Park, MD, June 1999.
- Seagull, A.B. and L.K. Schubert, "Smarter corpus-based syntactic disambiguation," TR 693, Computer Science Dept., U. Rochester, November 1998.
- Sikorski, T. and J. F. Allen, TRAINS-95 System Evaluation, TRAINS Tech. Note 96-3, Dept. of Computer Science, University of Rochester, 1996.
- Sikorski, T. and James Allen, "A Task-Based Evaluation of the TRAINS-95 Dialogue System," *Proceedings of the ECAI Workshop on Dialogue Processing in Spoken Language Systems*, August, 1996.
- Stent, Amanda and James F. Allen, "TRAINS-96 System Evaluation", TRAINS TN 97-1, Computer Science Dept., U. Rochester, March 1997.
- Tetreault, J.R., "Analysis of syntax based pronoun resolution methods," *Proc., 37th*

*Annual Meeting of the Assoc'n. for Computational Linguistics (ACL'99),
Student Session, College Park, MD, June 1999.*

Zaki, M.J., N. Lesh, and M. Ogihara, "PLANMINE: Predicting plan failures using sequence mining," TR 671, Computer Science Dept., U. Rochester, July 1998; to appear, *Artificial Intelligence Review* (Special Issue on the Application of Data Mining), 1999.

Zaki, M.J., N. Lesh, and M. Ogihara, "PLANMINE: Sequence mining for plan failures," *4th Int'l. Conf. on Knowledge Discovery and Data Mining (KDD)*, 369–373, New York, NY, August 1998.

Appendix A: Dialogue Systems—From Theory to Practice

James F. Allen, George Ferguson, Bradford W. Miller, Eric K. Ringger, Teresa Sikorski

Dept. of Computer Science
University of Rochester
Rochester, NY 14627

{james, ferguson, miller, ringger, sikorski}@cs.rochester.edu
<http://www.cs.rochester.edu/research/trains>

Abstract

We describe our work on building end-to-end spoken dialogue system that can interact with a user to solve simple planning problems. Since the goal of this work was to produce a working system, great emphasis was made on finding techniques that are robust against errors and lack of coverage within any single component. Robustness is achieved by a combination of statistical post-correction of speech recognition errors, syntactically- and semantically-driven robust parsing, and extensive use of the dialogue context. An additional constraint is that the techniques used must work in near real-time, otherwise there is no possibility of conducting actual dialogues with users. While full working systems are difficult to construct, they do open the doors to end-to-end evaluation. We present an evaluation of the system using time-to-completion and the quality of the final solution that suggests that most native speakers of English can use the system successfully with virtually no training.

A.1. Introduction

While there has been much research on dialogue, there have been very few actual dialogue systems. This reveals that most theories of dialogue abstract away many problems that must be faced in order to construct robust systems. Given well over twenty years of research in this area, if we can't construct a robust system even in a trivially simple domain, that bodes ill for progress in the field. Without working systems, we are very limited in how we can evaluate the worth of various theories and models.

The prime goal of the work reported here was to demonstrate that it is feasible to construct robust spoken natural dialogue systems. Thus we were not seeking to develop new theories of dialogue, but rather to develop techniques to enable existing theories to be applied in practice. We chose a domain and task that was as simple as possible yet couldn't be solved without the collaboration of the human and system. In addition, there were three fundamental requirements on the system:

- it must run in near real-time;
- the user should need minimal training and not be constrained in what can be said; and
- the dialogue should result in something that can be independently evaluated.

At the start of this experiment in November 1994, we had no idea whether it was possible. While re-

searchers were reporting good accuracy (95%) for speech systems in simple question-answering tasks, our domain was considerably different and required a much more spontaneous form of interaction. We also knew that it would not be possible to directly use general models of plan recognition to aid in speech act interpretation (as in Allen & Perrault, 1980, Litman & Allen 1987, Carberry 1990), as these models would not be anywhere close to real time. Similarly, it would not be feasible to use general planning models for the system back end or for planning its responses. We could not, on the other hand, completely abandon the ideas underlying the plan-based approach, as we knew of no other theory that could provide an account for the interactions. Our approach was to try to retain the overall structure of plan-based systems, but to use domain-specific reasoning techniques to provide real-time performance.

Dialogue systems are notoriously hard to evaluate as there is no well-defined “correct answer”. So we cannot give accuracy measures as is typically done to measure the performance of speech recognition systems and parsing systems. This is especially true when evaluating dialogue robustness, which results from many different sources including attempts to correct speech recognition errors, to use semantic knowledge to interpret fragments, and to use dialogue strategies that kept the dialogue flowing efficiently despite recognition and interpretation errors.

The approach we take is to use task-based evaluation. We measure how well the system does at helping the user solve the problem. The two most telling measures are time-to-completion and the quality of the final solution. In the evaluation described later in this paper, we show that all our subjects were able to use TRAINS to solve problems with only minimal training. We also evaluated the overall effectiveness of our robust processing techniques by comparing spoken dialogues with keyboard dialogues by the same subjects. Even with a 30% word error rate, speech turned out to be considerably more efficient than keyboard.

A.2. The Task and Example Session

The domain in TRAINS is simple route planning. The user is given a map on a screen showing cities, routes, and the locations of a set of trains (see Figure 1), and a verbal specification of a set of destination cities where trains are needed. The task is to find the most efficient set of routes possible. The route planner used by the system is deliberately weak so that interaction is needed to find good plans. Specifically, the planner cannot find routes longer than four hops without an intermediate city, and when it can generate a route, it randomly selects among the possibilities.

Typically, the examples in papers on dialogue systems are carefully hand-constructed and designed to show off specific system capabilities. Such examples, however, give little feeling for how robust the system is. Much more revealing is how the system responds to unplanned interactions with naive users, and how it interacts when things start to go wrong. As a result, we won't present any examples here in which everything went smoothly. Rather, we present dialogues that stress the system's capabilities, better illuminating what works and what areas require further improvement.

The dialogue below was with a non-computer person who had used the system a few times before. In the transcripts, we show the input as it was produced by the speech recognizer in ALL CAPS followed by

what was actually said. If the recognition was perfect we just show the speech recognition output. The dialogue contains a significant number of errors and has a 30% word error rate. The system display is also shown at key points¹. In this problem, the engines start at Detroit, Montreal and Albany, and the goal is to get trains to Milwaukee, Lexington and Washington.

- S: Ok. I think I'm ready to start.
 U: OKAY LET'S SEE CONTAIN FROM DETROIT TO WASHINGTON
Okay. Let's take a train from Detroit to Washington.
 S: What route would you like to get from Detroit to Washington?
 U: LET'S GO VIA TOLEDO AND PITTSBURGH
 S: (displays Figure 2) The terminal at city Scranton is delaying traffic due to localized heavy winds. The terminal at city Baltimore is delaying traffic due to localized heavy winds. An additional five hours will be needed to travel through them due to decreased visibility.
 U: NOW LET'S TAKE THE TRAIN FROM DETROIT TO WASHINGTON D S_X ANY
No. Let's take the train from Detroit to Washington via Cincinnati
 S: (displays Figure 3) Okay.
 U: OKAY IT THAT'S OKAY NOW
Okay that that's okay now
 S: Pardon Me?
 U: UP OKAY NOW LET'S TAKE THE TRAIN FROM MONTREAL TO LEXINGTON
Okay. Now let's take the train from Montreal to Lexington.
 S: I need help choosing a route from Montreal to Lexington.
 U: LET'S GO P_M TO TRY
Let's go via Detroit
 S: (displays a route to Detroit but not on to Lexington) Is this OK?
 U: YES NOW LET'S GO TO LEXINGTON
 S: Yep.
 U: OKAY NOW ADD STATE THE LAST TRAIN AND GO FROM ALBANY TO O_O'S
Okay Now let's take the last train and go from Albany to Milwaukee.
 S: No. What city are you trying to have the engine at Montreal arrive at, if I caught your drift. I don't understand where we are sending the engine at Albany to.
 U: ME A JET ADD ALBANY NEEDS TO GO TO MILWAUKEE
The engine at Albany needs to go to Milwaukee.
 S: Tell me a route to use to get from Albany to Milwaukee, please.
 U: I'D GO VIA BUFFALO
 S: (displays a route through Toronto and Buffalo) Yes.
 U: GO B_X SYRACUSE AT BUFFALO
Go via Syracuse and Buffalo.
 S: (displays Figure 4) yeah.
 U: THAT'S COULD I CAN
That's great. I'm done
 S: Hey, its the programming.
 U: I_NEED DONE
I'm done

A.3. Background

The TRAINS system builds on a wide range of previous work that has investigated the use of plan-based models for natural language understanding and generation. This

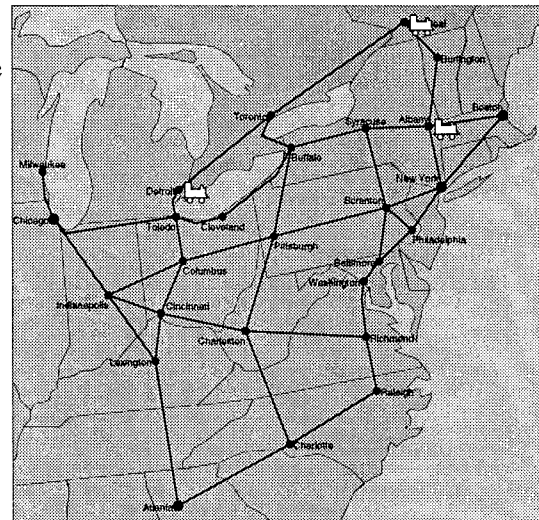


Figure 1: The initial scenario

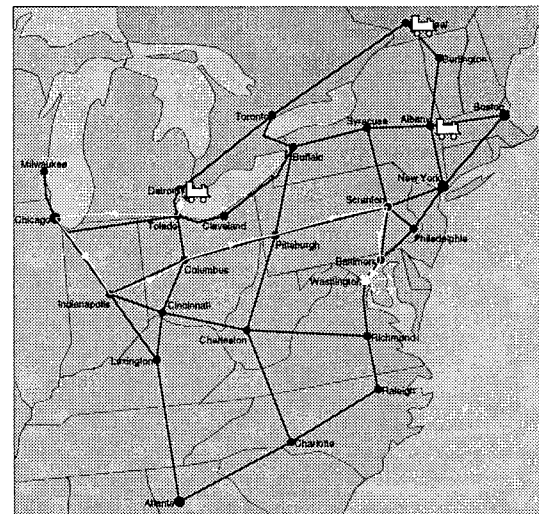


Figure 2: The proposed route

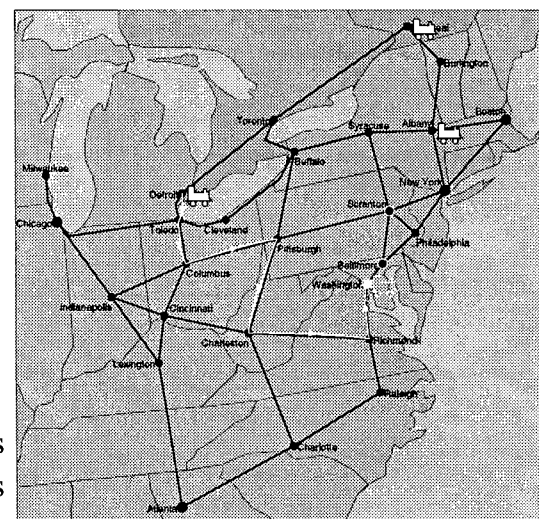


Figure 3: The corrected route

¹ A quicktime movie can be found as TRAINS95-v1.3-Pia.qt.gz at <http://www.cs.rochester.edu/research/trains/clips>.

section explores some of the ideas that had the greatest influence on the system.

The fundamental assumptions underlying the approach were developed in the speech act tradition in Philosophy. The focus of this work is on the communicative actions (or speech acts) that are performed when a person speaks. Austin (1957) introduced many of the basic arguments for this viewpoint, and showed how utterances in language can be best analyzed as actions such as requesting, promising, asserting, warning, congratulating, and so on. About the same time, Grice (1957) suggested the importance of intention recognition in communication, and argued that communication successfully occurs only when

the speaker has an intention to communicate and intends the hearer to recognize these intentions as part of the understanding process. Grice (1975) later pointed out that in order to recognize these intentions, the conversants must assume they are cooperating to the extent that what is said is relevant and possible to interpret in the current context. Searle (1969, 1975) combined the ideas of Austin and Grice and developed a highly influential theory of speech acts based on the notion of intention recognition.

Cohen & Perrault (1979) developed a computational model of speech acts as planning operators. Speech acts had preconditions and effects defined in terms of the intentions and beliefs of the speaker and hearer; e.g. an INFORM is a statement uttered in order to get the hearer to believe that a certain fact is true. A key advantage was that this approach allowed models to connect non-linguistic goals to linguistic actions. For instance, if Abe wants to open a safe but doesn't know the combination, and he believes Bea does know the combination, then he might reason (roughly) as follows: to open the safe I need to know the combination. I would know the combination if Bea told me. She could do this since she knows the combination, so I just have to get her to want to do the action, which I can do by asking her to do it. This quick summary avoids many subtleties, but gives a feel for the approach. The plan is summarized in Figure 5. Note that there is an alternate way to get the safe open, namely to ask Bea to open in. With

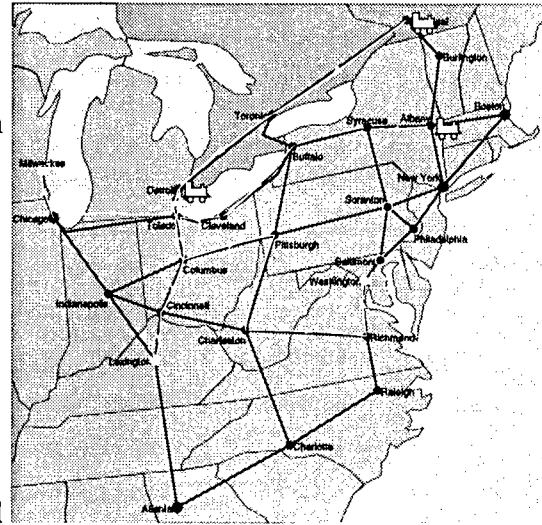


Figure 4: The final routes

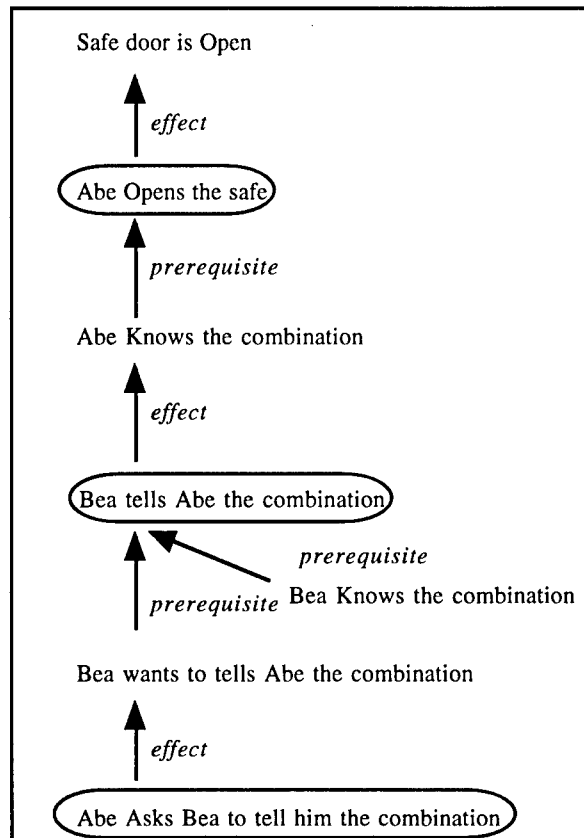


Figure 5: Planning to open the safe

this plan, the goal would still be achieved but Abe would not end up knowing the combination. The plan-based model naturally accounts for such variations.

Cohen and Perrault's model accounts for how speech acts could be planned to accomplish the goals of an agent. Allen and Perrault (1981) then developed a model of language understanding based on intention recognition, for which they developed a formalism for plan inference (or plan recognition). Roughly speaking, this approach reversed the planning process: rather than working from a goal to the actions required to accomplish that goal, it worked from observed actions (the speech acts) and identified plausible goals that might have motivated the actions.

One particular difficulty for speech act interpretation is that speakers frequently speak indirectly. For instance, rather than explicitly asking someone for the time, people typically ask indirectly by saying "Do you know the time?" or "Do you have a watch?". Perrault and Allen (1980) showed how the plan-based model could account for a wide range of indirect speech acts. Perhaps most important for our current work is that they proposed a model of a conversational agent: the agent observes surface speech acts (computed by the parser from utterances), uses plan recognition to identify underlying intentions, uses an "obstacle detection" process to identify problems in the user's plan and selects goals to address which then leads to planning to achieve these goals, which ultimately leads to speech acts as a response. This model is significantly different than that used in most NL systems, for they do not explicitly reason about the user's intentions at all. For example, in a typical question answering system, the response is based on the results of a query computed from the structure of the question. In the conversational agent model, the answer is determined from an analysis of the user's goals. This allows the system much more flexibility to recognize and address misconceptions, and to provide additional information that was not explicitly requested but that would be useful to the questioner, and to maintain the evolving context of the dialogue.

The speech act work provided a rich model for interpreting individual utterances, but did not account for the more global structure of dialogue, or discourse in general. Plan-based models have also been used to great advantage in modeling the global structure as well. This is especially true for the class of dialogues called task-oriented dialogues, in which the participants are communicating to accomplish some well-defined purpose. While focusing on task-oriented dialogues might seem a limitation, it is worth noting that almost any application one can think of for human-computer dialogue falls into this category. This can be seen in the range of tasks that have been the focus of research, such as planning routes as in the TRAINS system, to selecting furniture for an apartment subject to budgetary and other constraints, to planning student course schedules, to assembling devices such as water pumps, to finding out information about flights or train schedules, and to a wide range of other customer service agents (e.g., bank inquiries, catalog

- A.1 I do not have a sofa for a better price
- A.2 But I do have a floor-lamp, blue (\$250)
- A.3 I have a green table (\$200) and
- A.4 a chair for 75 dollars.
- A.5 Sorry I am taking so much time.
- A.6 I lost a chair.
- A.7 Meghan is finding it.
- B.8 Not a problem with the time
- B.9 Sorry about the typo
- B.10 My brain forgets that my fingers don't
function as quick as it does
- B.11 The lamp and table sound good
- B.12 But the chair seems expensive

Figure 6: A subdialogue

ordering, automated telephone operators, and automated travel agents).

Grosz (1974, 1977) showed that the flow of topic in a task-oriented dialog has a systematic relationship to the structure of the task. It is not necessarily identical to the task, however, as not all aspects of a task need be discussed, and the order of discussion does not necessarily have to follow exactly the order that steps would be executed. She argued that discourse is hierarchically organized, in which the flow of topics can be hierarchically organized in relation to the hierarchy in the task domain. Furthermore, a stack-like structure allows topics to be “suspended” and later “resumed”. This organization suggests a significantly different organization for processes such as reference resolution which are often simply based on recency; i.e., a definite noun phrase tends to refer to the object most recently mentioned that has the required characteristics. Grosz showed some counter examples to this that could be easily explained within the hierarchical model.

Grosz & Sidner (1986) refined these ideas with a theory that separately modelled the **intentional state** and the **attentional state**. The intentional state models discourse intentions (which they call **discourse segment purposes**) which captures the communicative intentions in an utterance and are distinct from the task-related intentions. The attentional state is a stack-like structure and is critical for defining the context for referential processing and disambiguation. For example, consider an example shown in Figure 6, which is a simplified version of an actual keyboard dialogue collected in the COCONUT project (1977). The sentences typed by A are marked as A.1, A.2, etc, and those typed by B with B.8, etc.. Note that in A.4, A suggests a chair. In B.12, B refers to the same chair to say that it seems expensive. But note that there was another chair more recently mentioned in A.6, but this does not seem to be considered. One reason for this is that A.5 through B.10 is a subdialogue that is addressing the interaction rather than the task. Discussion of the task resumes in B.11, and B.11 and B.12 are interpreted with respect to the context created by A.1 through A.4. This interpretation is further reinforced by considering what the discourse purposes are. A.2 through A.4 are clearly proposals for objects to purchase, and B.11 and B.12 are the responses to those proposals. The purposes underlying A.5 through B.10, on the other hand, are more to do with maintaining a sense of collaboration and cooperation between the participants.

Grosz & Sidner (1990) investigated the nature of discourse purposes in more detail, and argue that they require a theory of **Shared Plans**, plans that are developed collaboratively by the two agents, and not by any single agent. Clark (1996) explores similar ideas and argues that the interpretations of utterances are subject to negotiation by all the participants and not just what the speaker originally intended, but rather what the speaker and hearer eventually agree on what the interpretation was. A computational model that uses these notions for reference resolution was developed by Heeman & Hirst (1995).

In the work described so far, plans have been used in two different roles. In the speech act modeling and with discourse intentions, plans are used to model the communicative actions. In modeling topic flow, plans are used to model the task that the agents are performing. This suggests that multiple levels of plan reasoning may be required to fully model what is going on in a dialogue. Litman and Allen (1987, 1990) developed such a multi-level model which performed plan recognition simultaneously at

two levels in order to determine an interpretation. One level was the **domain level**, in which the actual actions being discussed are modeled (e.g., in TRAINS, these are actions such as moving trains), and the other was called the **discourse level**, which included communicative acts such as the speech acts with problem-solving related acts such as INTRODUCE-PLAN, IDENTIFY-PARAMETER, and so on. For example, consider a dialogue that starts as follows:

A.1 I want to get a train to Avon.

A.2 Let's use the engine at Bath.

Utterance A.1 would be recognized as an INTRODUCE-GOAL at the discourse level, and would cause the creation of a domain level plan with the goal of getting a train to Avon. Depending on the nature of the domain plan reasoning algorithm this might introduce an abstract action to move a train from somewhere to Avon. Utterance A.2 would then be recognized as an IDENTIFY-PARAMETER of the current domain plan. For this interpretation to make sense, the domain reasoner would need to find a way to fit the engine described into the plan. It could do this by instantiating the action of moving the engine at Bath to Avon. If this interpretation is accepted by the system, it would then "execute" the IDENTIFY-PARAMETER action which would modify the domain plan. This is summarized in Figure 7.

Utterance	Discourse Plan Recognized	Resulting Domain Plan
A.1 I want to get a train to Avon	INTRODUCE-PLAN <some engine> AT Avon	Goal: <some engin> AT Avon Plan: Move <some engine> to Avon
A.2 Let's use the engine at Bath	IDENTIFY-PARAMETER <some engine> = ENG1	Goal: ENG1 AT Avon Plan: Move ENG1 to Avon

Figure 7: Multi-level Plan Interpretation

One of the key ideas in this work was that one can model "discourse level" clarifications and corrections by introducing another discourse level plan that explicitly considers and modifies the previous discourse level plan. Carberry (1990) and others have refined this model further, developing a modified version in which the actions are broken into three levels: the domain, the problem-solving and the communication levels.

Despite the promise of all this work, when we started there had still not been a single robust dialogue system constructed using these principles. It is informative to consider why this was the case and then how we try to avoid these problems in the TRAINS system. There are at least four significant problems in using plan-based models in actual systems:

1. The knowledge representation problem: We have no fully adequate representational models of beliefs, intentions and plans that account for all known problematic cases;
2. The knowledge engineering problem: Defining the information needed to cover the full range of situations and possible utterances in a dialogue for a realistic task is difficult and time consuming;
3. The computational complexity problem: Known plan recognition and planning algorithms are too inefficient to run in close to real time; and

4. The noisy input problem: Speech recognition still involves a high word error rate on spontaneous speech, causing significant problems in parsing and interpretation.

So these were the problems we faced. For the most part, people continuing to work in this area have chosen to focus on the knowledge representation problem, and have developed better representational models for intentions and speech acts (e.g., Cohen & Levesque 1990a, b) and for shared plans (Grosz & Kraus, 1996). We addressed these problems from a different point of view. Rather than working to solve the full problem, we decided to design a limited domain, and then focus only on the mechanisms that were required to produce a robust system in that domain. Thus we are attacking the problems in a bottom-up rather than a top-down fashion. Critical to the success of this approach was the choice of a suitable domain. One too complex would make the task impossible, and one too easy would make the results not useful for subsequent work. Specifically, we had the following criteria for the domain:

- The domain should support a range of problems for people to solve, from trivial to challenging;
- The domain should require as few representational constructs as possible to minimize the amount of explicit knowledge that needs to be encoded;
- The domain should support a wide range of linguistic constructs so as to produce a rich set of dialogue behavior; and
- The domain should be intuitive so that a person can work in the domain and use the system with virtually no training.

The route planning task in TRAINS fits these requirements quite well. It supports a wide range of problems from the trivial, such as finding a route for one train, to complex problems in scheduling many trains where the routes interact with each other. Even with the complex problems, however, the domain requires very few representational primitives to encode the domain. Basically, in TRAINS we need to represent locations (e.g., cities), connections (e.g., routes), simple objects (e.g., trains), and the location of all these objects over time. Furthermore, there are only two actions: GO from one location to another, and STAY at a location for some period of time. Some simple experiments using two people solving problems in this domain verified that it was rich enough to generate a wide variety of linguistic constructions and dialogues, while still remaining considerably simplified from full language. The domain naturally limits the speakers so that no training is required regarding how to speak to the system. Finally, the domain is quite intuitive and people can learn the task after seeing or performing a single session.

Choosing the TRAINS domain goes a long way towards avoiding several of the problems above, as the significant aspects of the entire TRAINS world can be encoded using only a few predicates. Of course, these are only the “domain-level” predicates, and we still need to represent the discourse level actions and mental states of the participants. We handle this problem by making some radical simplifying assumptions, which only can be supported experimentally by observing that these assumptions have not yet caused any problems in the system. The most radical is that we do not model the individual beliefs of the hearer. Rather, we have what can be thought of as a “shared” set of beliefs which includes all the information that has been conveyed in the dialogue (by language or the GUI interface), and then private information that only the system currently knows. Some of this is explicit knowledge

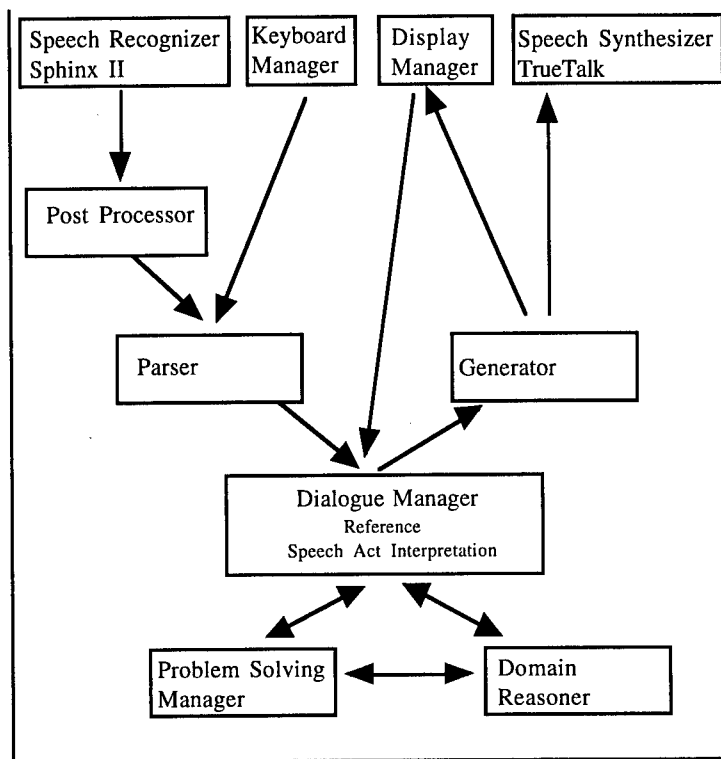


Figure 8: The System Architecture

(e.g., Avon is congested) while other information is discovered by the system's reasoning processes (e.g., two trains will be delayed because they are trying to use the same track at the same time).

As the plan is built, using the system's interpretations of suggestions from the user and the system's own suggestions based on its planning processes, all contributions end up as part of a single shared plan (which in TRAINS is always graphically displayed on the screen). Rather than trying to maintain separate versions of the plan (e.g., what S thinks the plan is, what S thinks the user thinks the plan is, and so on), a single plan is represented and we depend on the user or the system to offer corrections if the current plan is not to their liking. In many ways, this can be thought of as a primitive version of joint action models of discourse proposed by Clark (1996) and Heeman and Hirst (1995).

But having only a shared or joint plan does not eliminate the need for intention recognition. The system still needs to identify the user's intentions underlying an utterance in order to determine what changes the user is proposing (e.g., is "Go from Avon to Bath" a proposal of a new goal or a correction of an existing route?). And, on the other side of the coin, we haven't eliminated the need for the system to plan its own utterances. Thus we still have the computational complexity problem—there are no efficient plan recognition or planning algorithms, even in domains as simple as TRAINS. We avoid this problem using a combination of techniques. At the discourse level, we effectively "compile out" the recognition and planning process and drive the interpretation and response planning by a set of decision trees that ask questions that might have been generated from a plan-based discourse model. This decision tree depends heavily on syntactic clues to an initial interpretation which is then filtered and modified depending on how well the proposed operation fits in at the domain level.

Many decisions at the discourse level are reduced to questions about whether some operation makes sense at the domain level (e.g., does it make sense to modify the current route by avoiding Dansville). These questions are answered by a series of checks on constraints and preconditions of actions. For example, the system would reject a proposed modification of a route to avoid Dansville if that route did not currently go through Dansville. This uses a general principle that corrections implicitly define a condition on the current solution as well as identifying a property of the desired solution. For constraints of the form (AVOID Avon), the condition is that the current route goes through Avon. Note that this property does not hold for introducing or extending a route, and one can say “Go from Avon to Bath avoiding Corning” even though no route currently goes through Corning.

Of course, the system still eventually needs to do some planning at the domain level in order to find appropriate routes. This planning is performed by domain-specific reasoners. For instance, rather than planning routes from first principles using a general purpose planning algorithm, the system calls a fast route finding procedure and then converts its results into a sequence of actions.

In summary, planning search is replaced by specialized reasoning engines, and plan recognition is handled by decision trees that use constraint checking and plan reasoning (by the specialized reasoning engines). The other significant factor in solving the complexity problem is simply the faster machines that are available today. Even our fast methods would have been impractical just five years ago.

The noisy input problem will be the focus of much of this paper, as the presence of speech recognition errors greatly complicates all aspects of the system. Remember that it is not only that we miss words that were said, but that the speech recognizer will insert erroneous words in their places. We address this problem by correcting recognizer errors that have occurred consistently in the past and using robust interpretation rules tempered by the validation of possible interpretations in context (i.e., the intention recognition process). Robust rules occur at all levels: the parser operates bottom up so it can “skip” over uninterpretable input and build “islands of reliability”, semantic template matching is used to combine fragments into possible interpretations, speech acts are combined together at the discourse level to create better interpretations, intention recognition is used to attempt to identify the user’s intent even when the content is not fully determined, information is combined across utterances to identify plausible interpretation, and finally, dialogue strategies use a variety of clarification, confirmation and correction strategies to keep the dialogue running efficiently.

As a final comment on robustness, the strategies we used were developed bottom-up from direct experience with the system, rather than designed in advance. In particular, we built the initial prototype of the system in a three month period, and from then on ran sessions with users to identify what aspects of the system were actually causing dialogue failures. This allowed us to focus our activities. There were many things that we initially thought would be problematic but that turned out to be handled by relatively simple techniques within the TRAINS domain.

A.4. The System

The TRAINS system is organized as shown in Figure 8. At the top are the I/O facilities. The speech recognition system is the SPHINX-II system from CMU (Huang et al, 1993) with acoustic models trained from data in the Air Travel Information System (ATIS) domain. We have performed experiments using both the ATIS language model and a model constructed from TRAINS dialogue data. The speech synthesizer is a commercial product: the TRUETALK system from Entropic. The rest of the system was built at Rochester. The object-oriented Display Manager supports a communication language that allows other modules to control the contents of the display, and supports limited mouse-based interaction with the user. The speech recognition output is passed through the error-correcting post-processor described in section A.5. The parser, described in section A.6, accepts input either from the post-processor (for speech) or the keyboard manager (for typed input), and produces a set of speech act interpretations that are passed to the dialogue manager, described in section A.7. The Display Manager also generates speech acts when the user uses the mouse. The dialogue manager (DM) consists of subcomponents handling reference, speech act interpretation and speech act planning (the verbal reasoner). It communicates with the problem solving manager (PSM) to obtain information about the problem solving process and state of the plan so far, and with the domain reasoner for queries about the state of the world. When a speech act is planned for output, it is passed to the generator, which constructs a sentence to speak and/or a set of display commands to update the display.

The rest of this paper describes the major modules in more detail, including the post-processor, the parser, the dialogue manager and the problem solving manager. The generator will not be discussed further as it is currently a simple template-based system. It uses templates associated with different speech act forms that are instantiated with descriptions of the particular objects involved. The form of these descriptions is defined individually for each class of objects in the domain. While primitive, it did not have a significant effect on the system's performance of the task.

The remainder of the paper explores the main modules in more detail, focusing on how speech input is processed, through the post-processing, the parser to the dialogue manager, which uses the problems solving manager to evaluate interpretations and construct the plans.

A.5. Statistical Error Correction

The speech postprocessor (SPEECHPP) corrects speech recognition (SR) errors using a statistical model of the recognizer's past performance in the TRAINS domain. The following are examples of speech recognition errors that occurred in the sample dialogue. Appendix A1 contains the entire dialogue and the corrections attempted. In each, the words tagged REF indicate what was actually said, while those tagged with SR indicate what the speech recognition system proposed, and PP indicates the output of SPEECHPP. While the corrected transcriptions are not perfect, they are typically a better approximation of the actual utterance. As the first example shows, some recognition errors are simple word-for-word confusions:

```
SR:  GO B_X SYRACUSE AT BUFFALO
PP:  GO VIA SYRACUSE VIA BUFFALO
REF: GO VIA SYRACUSE AND BUFFALO
```

In the next example, a single word was replaced by more than one smaller word:

SR: LETS GO P_M TO TRY
PP: LETS GO P_M TO DETROIT
REF: LETS GO VIA DETROIT

The post-processor yields fewer errors by compensating for the mismatch between the language used to train the recognizer and the language actually used with the system. To achieve this, we adapted some techniques from statistical machine translation (such as Brown et al., 1990) in order to model the errors that Sphinx-II makes in our domain. Briefly, the model consists of two parts: a channel model, which accounts for errors made by the SR, and a language model that accounts for the likelihood of a sequence of words being uttered in the first place.

More precisely, given an observed word sequence O from the speech recognizer, SPEECHPP finds the most likely original word sequence S by finding the word sequence S that maximizes the expression $Prob(O|S) * Prob(S)$, where

- $Prob(S)$ is the probability that the user would utter sequence S , and
- $Prob(O|S)$ is the probability that the SR produces the sequence O when S was actually spoken.

For efficiency, it is necessary to estimate these distributions with relatively simple models by making independence assumptions. For $Prob(S)$, we train a word-bigram "back-off" language model (Katz, 87) from hand-transcribed dialogues previously collected with the TRAINS system. For $Prob(O|S)$, we build a channel model that incorporates a model of one-for-one as well as one-for-two and two-for-one substitutions. We say that the channel is "fertile" because of the possibility that words multiply in the channel. For more details, see Ringger and Allen (1996). The channel model used in the example dialogue assumed only single word-for-word substitutions.

The channel model is trained by automatically aligning the hand transcriptions with the output of Sphinx-II on the utterances in the (SPEECHPP) training set and by tabulating the confusions that occurred.

We use a Viterbi beam-search to find the best correction S that maximizes the expression. This search is the heart of our post-processing technique but is widely known so is not described here (see Forney (1973) and Lowerre (1986)).

Having a relatively small number of TRAINS dialogues for training, we wanted to investigate how well the data could be employed in models for both the SR and the SPEECHPP. We ran several experiments to weigh our options. For a baseline, we built a class-based back-off language model for Sphinx-II using only transcriptions of ATIS spoken utterances. Using this model, the performance of Sphinx-II alone was 58.7% on TRAINS-domain utterances. This is not necessarily an indictment of Sphinx-II but simply reflects the mismatched language models.

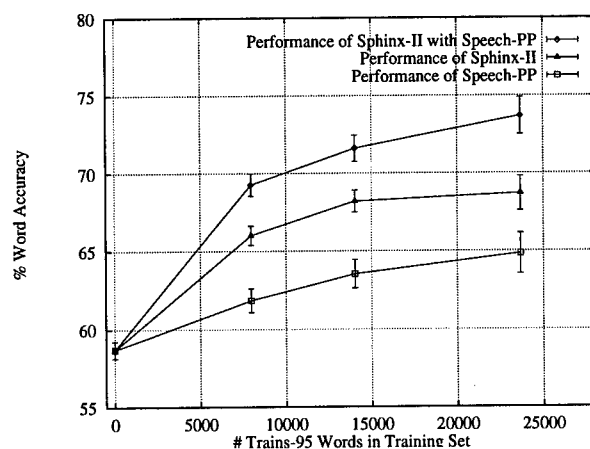


Figure 9: Post-processing Evaluation

From this baseline, we used varying amounts of training data exclusively for building models for the SPEECHPP; this scenario would be most relevant if the speech recognizer were a black-box and we did not know how to train its model(s). Second, we used varying amounts of the training data exclusively for augmenting the ATIS data to build a language model for Sphinx-II. Third, we combined the methods, using the training data both to extend the language model in Sphinx-II and to then train SPEECHPP on the newly trained system.

The results of the first experiment are shown by the second curve from the bottom of Figure 9, which indicates the performance of the SPEECHPP over the baseline Sphinx-II. The first point comes from using approximately 25% of the available training data in the SPEECHPP models. The second and third points come from using approximately 50% and 75%, respectively, of the available training data. The curve clearly indicates that the SPEECHPP does a reasonable job of boosting our word recognition rates over baseline Sphinx-II. We did not use all of our available data, since the remainder was used to determine the test results via repeated leave-one-out cross-validation.

Similarly, the results of the second experiment are shown by the middle curve. The points reflect the performance of Sphinx-II (without SPEECHPP) when using 25%, 50%, and 75% of the available training data in its language model. These results indicate that equivalent amounts of training data can be used with greater impact in the language model of the SR than in the post-processor.

Finally, the outcome of the third experiment is reflected in the uppermost curve. Each point indicates the performance of the SPEECHPP using a set of models trained on the behavior of Sphinx-II for the corresponding point from the second experiment. The results from this experiment indicate that even if the language model of the SR can be modified, then the post-processor trained on the same new data can still significantly improve word recognition accuracy. Hence, whether the SR's models are tunable or not, the post-processor is in neither case redundant.

A.5. Robust Parsing

Given that errors are inevitable, robust parsing techniques are essential. We use a pure bottom-up parser (using the algorithm described in (Allen, 1995)) in order to identify the possible constituents at any point in the utterance based on syntactic and semantic restrictions. Every constituent in each grammar rule specifies both a syntactic category and a semantic category, plus other features to encode co-occurrence restrictions as found in many grammars. The semantic features encode selectional restrictions, most of which are domain-independent. For example, in effect there is no general rule for PP attachment in the grammar. Rather there are rules for temporal adverbial modification (e.g., *at eight o'clock*), locational modification (e.g., *in Chicago*), and so on.

The end result of parsing is a sequence of speech acts rather than a syntactic structure. Viewing the output as a sequence of speech acts has significant impact on the form and style of the grammar. It forces an emphasis on encoding semantic and pragmatic features in the grammar. There are, for instance, numerous rules that encode specific conventional speech acts (e.g., *That's good* is a CONFIRM, *Okay* is an CONFIRM/ACKNOWLEDGE, *Let's go to Chicago* is a SUGGEST, and so on). Simply classifying such utterances as sentences would miss the point. Thus the parser computes a set of plausible speech act interpretations based on the surface form, similar to the model described in Hinkelman & Allen (1989).

We use a hierarchy of speech acts that encodes different levels of vagueness, with the root being a speech act that indicates content with an unspecified illocutionary force. This allows us to always have an illocutionary force identified, which can be refined as more of the utterance is processed. The final interpretation of an utterance is the sequence of speech acts that provides the “minimal covering” of the input - i.e., the shortest sequence that accounts for the input. If an utterance was completely uninterpretable, the parser would still produce an output, namely, a speech act with no specified illocutionary force or content! The speech act hierarchy in TRAINS is shown in Figure 10.

Figure 11 gives the logical form computed by the parser for the sentence “Okay now go from Avon to Bath please”. It includes a list of all objects mentioned in the utterances, the paths, as well as other information such as politeness cues, a reliability measure and some syntactic information. A fairly traditional logical form could be obtained by starting at the value of the SEMANTICS slot and chasing down the pointers through the objects in paths. The advantage of this distributed representation is that we obtain similar representation from fully parsed utterances as from utterances that consist only of fragments. This uniformity is crucial for subsequent robust processing.

Speech Act Hierarchy	Use	Example
SPEECH-ACT	uninterpretable utterances	"Us the go Avon"
TELL	simple declaratives and fragments	"Avon is congested"
ID-GOAL	utterance explicitly introduces a goal	"I want to take a train to Bath"
EVALUATION	declarative that evaluates an option	"That's good"
REQUEST	generic directive act for imperatives	"Go to Corning"
SUGGEST-ACTION	suggestions of actions	"Let's go to Corning"
YN-QUESTION	yes-no questions	"Is the train at Avon?"
WH-QUESTION	wh question	"Where are the trains?"
AGREEMENT		
ACCEPT/CONFIRM	accepting a proposal	"OK", "yes"
REJECT	rejecting a proposal	"No, that won't work"
CONVENTIONAL		
CLOSE	conventional closing	"good-bye"
GREET	conventional greeting	"Hi"
NOLO-COMPRENDEZ	signal non-comprehension	"huh?"
APOLOGIZE	apologizing	"I'm sorry"
NOLO-PROBLEMO	responding to an apology	"no problem"
ACKNOWLEDGE	acknowledgement of understanding	"OK", "uh-huh"

Figure 10: The Speech Act Hierarchy in TRAINS

```

(CONFIRM
:SEMANTICS OKAY
:RELIABILITY 100
:MODE :DISPLAY
:INPUT (OKAY))
(REQUEST
:OBJECTS ((DESCRIPTION (STATUS NAME) (VAR V2242) (CLASS CITY)
(LEX AVON) (SORT INDIVIDUAL))
(DESCRIPTION (STATUS NAME) (VAR V2242) (CLASS CITY)
(LEX BATH) (SORT INDIVIDUAL)))
:PATHS ((PATH (VAR V2238) (CONSTRAINT (AND (FROM V2238 V2242)
(TO V2238 V2253)))))
:SEMANTICS (PROP (VAR V2231) (CLASS GO-BY-PATH)
(CONSTRAINT (AND (LSUBJ V2231 *YOU*) (LOBJ V2231 V2238)))
:SOCIAL-CONTEXT (POLITENESS PLEASE)
:RELIABILITY 100
:MODE TEXT
:SYNTAX ((SUBJECT . *YOU*) (OBJECT))
:INPUT (GO FROM AVON TO BATH PLEASE))

```

Figure 11: The logical form for "Okay Go from Avon to Bath please"

Consider an example of how an utterance from the sample dialogue is parsed showing how speech recognition errors are handled. The input, after post processing, is "Okay now I take the last train in go from Albany to is". The best sequence of speech acts to cover this input consists of three acts:

1. a CONFIRM (“Okay”)
2. a TELL, with content to take the last train (“Now I take the last train”)
3. a REQUEST to go from Albany (“go from Albany”)

Note that the “to is” at the end of the utterance is simply ignored as it is uninterpretable. While not present in the output, the presence of unaccounted words will lower the parser’s confidence score that it assigns to the interpretation.

The actual utterance was “Okay now let’s take the last train and go from Albany to Milwaukee”. Note that while the parser is not able to reconstruct the complete intentions of the user, it has extracted enough to continue the dialogue in a reasonable fashion by invoking a clarification subdialogue. Specifically, it has correctly recognized the confirmation of the previous exchange (act 1), and recognized a request to move a train from Albany (act 3). Act 2 is an incorrect analysis, and results in the system generating a clarification question that the user ends up ignoring. Thus, as far as furthering the dialogue, the system has done reasonably well.

A.7. The Problem Solver and Domain Reasoner

The next phase of processing of an utterance is the interpretation by the dialogue manager. Before we consider this, however, we first examine the problem solving, as the dialogue manager depends heavily on the problem solving manager when interpreting utterances. The problem solver is called to evaluate all speech acts that appear to constrain, extend or change the current plan and then to perform the requested modification to the plan once an interpretation has been determined. It maintains a hierarchical representation of the plan developed so far. This is used to help focus in the interpretation process, as well as to organize the information needed to invoke the domain reasoner to reason about the plans under construction (in this case, routes).

Goals, Solutions and Plans

In traditional planning systems (e.g., Fikes & Nilsson, 1971, Weld, 1994), goals are often simply the unsatisfied preconditions or undeveloped actions in the hierarchy. Once a plan is fully developed, it is hard to distinguish between actions that are present because they are the goals and actions that were introduced because they seemed a reasonable solution to the goal. Losing this information makes many operations needed to support interactive planning very difficult. Consider plan revision, a critical part of interactive planning. If we can’t tell what the motivating goals were, it is difficult to determine what aspects of the plan should be kept and what can be changed. As a very simple example, consider two situations with identical solutions in the TRAINS domain: in the first case, the goal is to go from Toledo to Albany, and the solution is to take the route from Toledo, through Columbus, Pittsburgh, and Scranton. In the second case, the goal is to go from Toledo to Albany avoiding Buffalo, with the same route. Say the user wants to change the route in each case, and indicates this by a simple rejection. In the first case, the planner might suggest the route through Cleveland, Buffalo, and Syracuse, since it seems shorter than the original. In the second plan, though, this would be an inappropriate suggestion as it doesn’t meet the goal, which requires the route avoid Buffalo.

In TRAINS, the goal hierarchy captures the goals and subgoals that have been explicitly developed

through the interaction with the user. Particular solutions to the goals, on the other hand, are not in the hierarchy explicitly, although they are associated with the goals that they address. The goals provide an organization of the solution that is useful for many purposes - for revising parts of the plan, for summarizing the plan, and for identifying the focus of the plan for a particular interaction. For example, consider the problem solving state shown in Figure 12.

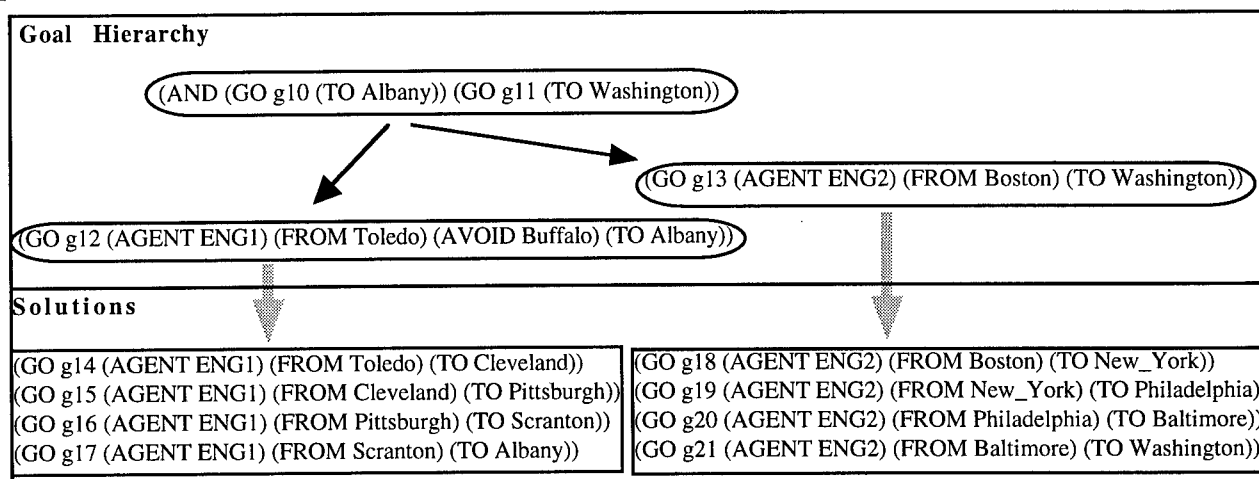


Figure 12: A Simple Problem Solving State

There are two subplans, which in this particular case are independent of each other as they involve different trains, cities and tracks. The planned actions for ENG1 satisfy the stated goal of going from Chicago to Albany avoiding Buffalo, while the planned actions for ENG2 satisfy the goal of going from Boston to Atlanta. If we needed an alternate solution to the first subgoal, the new solution should still avoid Buffalo. With the second goal, the only part that is crucial to the goal is the origin and destination.

The goal hierarchy plays a crucial role in managing the focus of attention during the interaction. In general, a user will move up and down the goal hierarchy in fairly predictable ways as they work on one problem and then move to the next. The routes are constructed by a specialized domain reasoner, whose results are then converted into a solution by the problem solving manager.

The Problem Solving Operations

From studying human-human interactions in joint problem solving (Heeman and Allen, 1995), and from our experience in building the TRAINS system, we have identified an initial set of problem solving interactions that characterize the operations that can be performed by the user through speech acts. These operations define an abstract domain-independent model of a problem solving system that can support mixed-initiative planning. Each operation is associated with a *focus node*, which indicates the subgoal currently under consideration as well as its associated solution if there is one. Sometimes an operation changes the goal, while other times it only changes the solution leaving the goals unchanged. The operations supported by the problem solving manager in TRAINS are specified in Figure 13.

NEW-SUBPLAN	Introduces a new goal (or subgoal) to be accomplished
REFINE	Refines a goal specification into a more specific goal
EXTEND	Extends a plan by adding an additional action or goal
MODIFY	Modifies an existing plan, removing some previous part of the plan
DO-WHAT-YOU-CAN	Attempts to interpret an action as a REFINe, EXTEND, or MODIFY operation
CANCEL	Removes a goal, subgoal, or object specified in the plan
CONFIRM	Verifies the structure and content of an interpretation
REJECT-SOLUTION	Rejects a current solution for a goal (and the system produces another)
SELECT-SOLUTION	Selects a specific solution out of a range of options
NEW-SCENARIO	Defines a new set of background assumptions for a new planning task
DELETE-PLAN	Deletes a specific plan structure from the plan hierarchy
UNDO	Backs up to a previous problem solving state

Figure 13: The Problem Solving Operations in TRAINS

The TRAINS problem solver implements the strategy of automatically looking for a solution whenever a new goal is introduced or modified (see Ferguson et al. 1996). Thus, to the user it is not always apparent whether an interaction changes the goals or just changes the solution. Recognizing which of these cases is appropriate, however, is crucial for the system, especially in determining how to interpret future interactions and revisions.

Figure 14 identifies the correct problem solving action and the problem solving action intended for each of the user's utterances in the sample dialogue. Note that this shows what the user actually said and what operation was intended rather than what the system actually did. In several cases, speech recognition errors prevented the system from identifying the correct intention. But each utterance is considered given the prior context in the dialogue, including the system responses based on errors up to that point.

Utterance as Actually Said	Intended Problem Solving Operation
Okay let's send a train from Detroit to Washington	NEW-SUBPLAN with goal <Go from Detroit to Washington>
Let's go via Toledo and Pittsburgh	REFINE producing goal <GO from Detroit to Washington via Toledo and Pittsburgh>
No Let's take the train from Detroit to Washington via Cincinnati	REJECT, MODIFY to produce goal <GO from Detroit to Washington via Cincinnati Avoiding Toledo and Pittsburgh>
Okay That's Okay now	CONFIRM goal <GO from Detroit to Washington via Cincinnati Avoiding Toledo and Pittsburgh>
Okay Now let's take the train from Montreal to Lexington	NEW-SUBPLAN with goal <GO from Montreal to Lexington>
Let's go via Detroit	REFINE producing goal <GO from Montreal to Lexington via Detroit>
Yes Now let's go to Lexington	EXTEND goal <GO from Montreal to Detroit> to <GO from Montreal to Lexington via Detroit>
Okay Now let's take the last train and go from Albany to Milwaukee	NEW-SUBPLAN with goal <GO from Albany to Milwaukee>

Figure 14: The intended Problem Solving Operations behind each Utterance

The problem solving manager provides a wide range of reasoning capabilities that both allow the dia-

logue manager to determine what speech acts make sense at any particular time in the dialogue, as well as providing the interface to the domain reasoners that then actually do the work required. We now turn to the dialogue manager, which determines the overall system's behavior.

A.8. Robust Discourse Processing

The dialogue manager is responsible for interpreting the speech acts in context, invoking the problem solving actions, formulating responses, and maintaining the system's idea of the state of the discourse. It maintains a discourse stack similar to the attentional state of Grosz & Sidner (1986). Each element of the discourse stack captures a context for interpreting utterances that consists of

1. the domain or discourse goal motivating the segment;
2. the focus node in the problem solving hierarchy;
3. the object focus and history list for the segment; and
4. information on the status of problem solving activity (e.g., has the goal been achieved yet or not).

Figure 15 shows a simple discourse stack consisting of two contexts, each pointing to a node in the problem solving hierarchy. Such a situation could arise if the participants were discussing how to get from Atlanta to Toronto, and then one suggested that they work on getting from Atlanta to Pittsburgh first. The subgoal now becomes the domain-level motivation in the dialogue, and the original goal remains on the stack indicating that it may be resumed at a later stage. The system always attempts to interpret an utterance with respect to the context that is at the top of the discourse stack. If no good interpretation is possible, it tries contexts lower in the stack, popping off the higher contexts. If no context produces a good interpretation, it calls the problem solving manager to allow it to attempt to construct a relevant context based on the goal hierarchy. Note that the problem solving hierarchy does not correspond to the intentional state of Grosz and Sidner as the hierarchy captures the domain-level goals, not the discourse segment purposes. <<finish>>

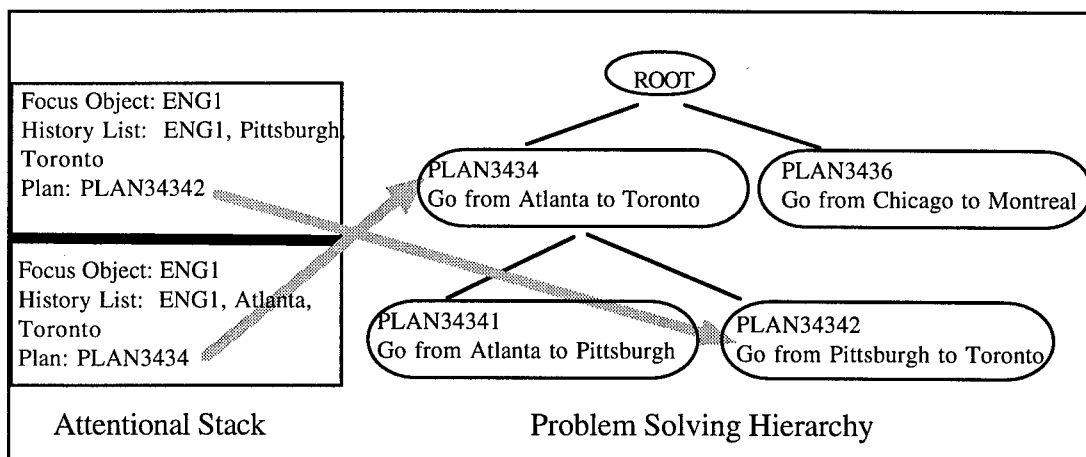


Figure 15: The Attentional Stack and its relation to the Problem Solving State

A fundamental principle in the design of TRAINS was a decision that, when faced with ambiguity it is better to choose a specific interpretation and run the risk of making a mistake as opposed to generating a clarification subdialogue. This occurs at several different levels. At the domain level, when a goal is

specified there are typically many different possible solutions. Rather than making the user specify enough information to uniquely select a solution, the system chooses one and presents it to the user for comment (Ferguson et al., 1996). At the discourse level, the system is often faced with uncertainty as to the intended operation, often due to recognition errors. In these cases, it often selects one of these interpretations and continues the dialogue rather than entering a clarification subdialogue. Of course, the success of this strategy depends on the system's ability to recognize and interpret subsequent corrections if they arise. Significant effort was made in the system to detect and handle a wide range of corrections, both in the grammar, the discourse processing and the domain reasoning. In later systems, we plan to specifically evaluate whether this decision was the correct one.

The discourse interpretation rules may reinterpret the parser's assignment of illocutionary force if it has additional information to draw upon. One way we attain robustness is by having overlapping realms of responsibility: one module may be able to do a better job resolving a problem because it has an alternative view of it. On the other hand, it's important to recognize another module's expertise as well. It could be disastrous to combine two speech acts that arise from "I really <garbled> think that's good", for instance, since the garbled part may include the word "don't". Since speech recognition may substitute important words one for the other, it's important to keep in mind that speech acts that have no firm illocutionary force due to the lack of a full parse may have little to do with what the speaker actually said. On the other hand, simply not processing such utterances would cripple the interaction. To address this dilemma, we allow a range of robustness strategies and depend heavily on contextual interpretation to evaluate whether proposed operations make sense in context.

The verbal reasoner is organized as a set of prioritized rules that match patterns in the input speech acts and the discourse state. These rules allow robust processing in the face of partial or ill-formed input as they match at varying levels of specificity, including rules that interpret fragments that have no identified illocutionary force. For instance, one rule would allow a fragment such as "to Avon" to be interpreted as a suggestion to extend a route, or an identification of a new goal. The prioritized rules are used in turn until an acceptable result is obtained.

As an example of the discourse processing, consider how the system handles the user's first utterance in the dialogue, "Okay let's send contain from Detroit to Washington". From the parser we get three acts:

1. a CONFIRM/ACKNOWLEDGE (*Okay*)
2. a TELL involving some uninterpretable words (*Let's send contain*)
3. a TELL act that mentions a route (*from Detroit to Washington*)

The dialogue manager sets up its initial conversational state and passes the act to reference for identification of particular objects, and then hands the acts to the verbal reasoner. Because there is nothing on the discourse stack, the initial confirm has no effect. (Had there been something on the stack, e.g. say the system had just asked a question, then the initial act might have been taken as an answer to the question, or if the system had just suggested a route, it would be taken as a confirmation). The following empty TELL is uninterpretable and hence ignored. While it is possible to claim the "send" could be used to indicate the illocutionary force of the following fragment, and that a "container"

might even be involved, the fact that the parser separated out the speech act indicates there may have been other fragments lost. The last speech act could be a suggestion of a new goal to move from Detroit to Washington. After checking that there is an engine at Detroit, this interpretation is accepted. In the version of the system used for this dialogue, the planner was limited to finding routes involving fewer than four hops, thus it was unable to generate a path between these cities. It returns two items:

1. an identification of the speech act as a suggestion of a goal to take a train from Detroit to Washington
2. a signal that it couldn't find a solution to satisfy the goal

The discourse context is updated and the verbal reasoner generates a response to clarify the route desired, which is realized in the system's response: "What route would you like to get from Detroit to Washington?".

As another example of robust processing, consider an interaction later in the dialogue in which the user's response "no" is misheard as "now": "Now let's take the train from Detroit to Washington do S_X Albany" (instead of "No let's take the train from Detroit to Washington via Cincinnati"). Since no explicit rejection is identified because of the recognition error, this utterance looks like a confirm and continuation on with the plan. Thus the verbal reasoner calls the problem solver to extend the path with the currently focused engine (say *engine1*) from Detroit to Washington.

The problem solver realizes that *engine1* isn't currently in Detroit, so this can't be a route extension. In addition, there is no other engine at Detroit, so this is not plausible as a focus shift to a different engine. Since *engine1* originated in Detroit, the verbal reasoner then decides to reinterpret the utterance as a correction. Since the utterance adds no new constraints, but there are the cities that were just mentioned as having delays, it presumes the user is attempting to avoid them, and invokes the domain reasoner to plan a new route avoiding the congested cities. The new path is returned and presented to the user.

While the response does not address the user's intention to go through Cincinnati due to the speech recognition errors, it is a reasonable response to the problem the user is trying to solve. In fact, the user decides to accept the proposed route and forget about going through Cincinnati. In other cases, the user might persevere and continue with another correction such as *No, through Cincinnati*. Robustness arises in the example because the system uses its knowledge of the domain to produce a reasonable response. The primary goal of the discourse system, namely to contribute to the overall goal of helping the user solve a problem with a minimum number of interactions, is best served by the "strong commitment" model; typically it's easier to correct a poor plan, than having to keep trying to specify a perfect one, particularly in the face of recognition problems.

A.9. Evaluating the System

While examples can be illuminating, they really don't address the issue of how well the system works overall. That can only be demonstrated by controlled experimentation. To give an idea of this, consider an evaluation that we performed on TRAINS-95, an earlier version of the system described here. To explore how well the system robustly handles spoken dialogue, we designed an experiment to contrast speech input with keyboard input. The experiment evaluated the effect of input medium on

the ability to perform the task and explored user input medium preferences. Task performance was evaluated in terms of two metrics: the amount of time taken to arrive at a solution and the quality of the solution. Solution quality for our domain is determined by the amount of time needed to travel the planned routes.

Sixteen subjects for the experiment were recruited from undergraduate computer science courses. None of the subjects had ever used the system before. The procedure was as follows:

- The subject viewed an online tutorial lasting 2.4 minutes that explained the mechanics of using the system but did not explain the capabilities of the system or train the user on how to interact with it.
- The subject was then allowed a few minutes to practice speech in order to allow them to adapt to the speech recognizer.
- All subjects were given identical sets of 5 tasks to perform, in the same order. Half of the subjects were asked to use speech first, keyboard second, speech third and keyboard fourth. The other half used keyboard first and then alternated. All subjects were given a choice of whether to use speech or keyboard input to accomplish the final task.
- After performing the final task, the subject completed a questionnaire.

The tasks were of similar complexity to the task solved in the example dialogue. Each involved three trains, and in each situation there was at least one delay affecting what looks like a reasonable route.

An analysis of the experiment results shows that the plans generated when speech input was used are of similar quality to those generated when keyboard input was used. However, the time needed to develop plans was significantly lower when speech input was used. In fact, overall, problems were solved using speech in 68% of the time needed to solve them using the keyboard. Figure 16 shows the task completion time results, and Figure 17 gives the solution quality results, each broken out by task.

Of the 16 subjects, 12 selected speech as the input mode for the final task and 4 selected keyboard input. Three of the four selecting keyboard input had actually experienced better or similar performance using keyboard input during the first four tasks. The fourth subject indicated on his questionnaire that he believed he could solve the problem more quickly using the keyboard; however, that subject had solved the two tasks using speech input 19% faster than the two tasks he solved using keyboard input.

Of the 80 tasks attempted, there were 7 in which the stated goals were not met. In each unsuccessful attempt, the subject was using speech input. There was no particular task that was troublesome and no particular subject that had difficulty. Seven different sub-

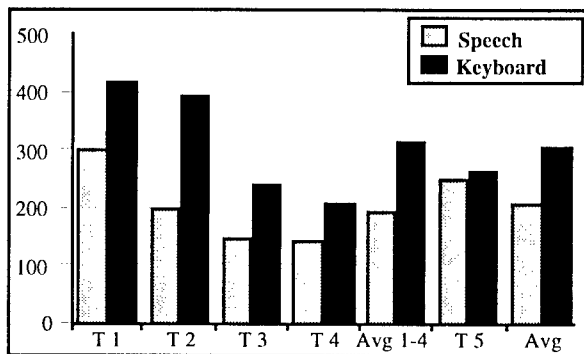


Figure 16: Time to Completion by Task

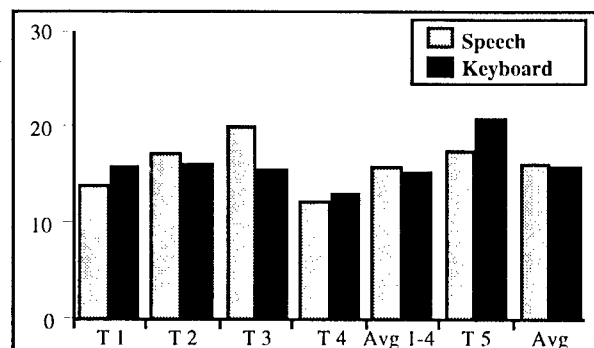


Figure 17: Length of Solution by Task

jects had a task where the goals were not met, and each of the five tasks was left unaccomplished at least once.

A review of the transcripts for the unsuccessful attempts revealed that in three of the seven failure cases, the subject misinterpreted the system's actions, and ended the dialogue believing the goals were met. Each of the other four unsuccessful attempts resulted from a common sequence of events. After the system proposed an inefficient route, word recognition errors caused the system to misinterpret rejection of the proposed route as acceptance. The subsequent subdialogues intended to improve the route were interpreted to be extensions to the route, causing the route to "overshoot" the intended destination. This analysis led us to redesign the problem solving component of the system, to the version described in this paper, so that the system could better identify when the user returned to a previously completed route in order to modify it.

A.9. Discussion

We know of no previous dialogue system that can be successfully used by untrained users to perform a concrete task as complex as TRAINS problems. Several speech systems, such as PEGASUS (Zue et al, 1994), WHEELS (Meng et al, 1996), and RAILTEL (Bennacef et al, 1996) support some dialogue-style interactions but don't involve explicit reasoning about the domain to perform their tasks (i.e., the task being performed can be hard-coded in advance and never changes during the dialogue). The AGS demonstrator (Sadek et al, 1996) is notable in that it uses a general model of a dialogue agent to drive the dialogue, but like the other systems above does not require any explicit domain reasoning to perform the task. One system that does perform explicit domain reasoning is the Duke system (Smith et al, 1994, 1995), which has a sophisticated domain reasoner and uses an integrated problem solving component to drive the dialogue. However, the system uses a highly restricted subset of English and requires significant training to use. Because of the scalability of the domain, TRAINS provides a unique test bed for evaluating the research in spoken dialogue systems.

Our research strategy was bottom-up with inspiration from the "top-down" theories in previous work. We have attempted to build a fully functional system in the simplest domain possible, and now plan to increase the complexity of the domain and focus on the problems that most significantly degrade overall performance. This leaves us open to the criticism that we are not using the most sophisticated models available. In an extreme case, consider our generation strategy. Template-based generation is not new and is clearly inadequate for more complex generation tasks. In fact, when starting the project we thought generation would be a major problem. However, the problems we expected have not arisen. While we could clearly improve the output of the system even in this small domain, the current generator does not appear to drag the system's performance down. We tried the simplest approaches first and then only generalized those algorithms whose inadequacies clearly degrades the performance of the system.

Likewise, we view the evaluation as only a very preliminary first step. While our evaluation might look like we're addressing an HCI issue of whether speech or keyboard is a more effective interface, this comparison was not our goal. Rather, we used the modality switch as a way of manipulating the

error rate and the degree of spontaneous speech. While keyboard performance is not perfect because of typos (we had a 5% word error rate on keyboard), it is considerably less error prone than speech. All we conclude from this experiment is that our robust processing techniques are sufficiently good that speech is a viable interface in such tasks even with high word error rates. In fact, it appears to be more efficient than the keyboard equivalent. Furthermore, subjects preferred the speech interface when given the choice (cf. Rudnicky, 1993).

Despite the limitations of the current evaluation, we are encouraged by this first step. It seems obvious to us that progress in dialogue systems is intimately tied to finding suitable evaluation measures. And task-based evaluation seems one of the most promising candidates, since it measures the impact of a proposed technique on the effectiveness of the interaction, rather than an abstract accuracy figure that may or may not have any effect on the overall functionality and effectiveness of the system.

Another area where we are open to criticism is that we used algorithms specific to the domain in order to produce effective intention recognition, disambiguation, and domain planning. Thus, the success of the system may be a result of the domain and say little about the plan-based approach to dialogue. To be honest, with the current system, it is hard to defend ourselves against this. This is a first step in what we see as a long ongoing process. To look at it another way: if we couldn't successfully build a system by employing whatever we could, then there is little hope for finding effective more general solutions. More generally, however, we believe that domain-specific knowledge will always be necessary in order to efficiently solve realistically complex problems. Our work on the problem solver in TRAINS is directed towards finding principled ways of incorporating such knowledge.

While there are clearly many places in which our system requires further work, it does set a new standard for spoken dialogue systems. More importantly, it will allow us to address a large number of research issues in the future in a much more systematic way, supported by empirical evaluation.

Acknowledgements

This work was supported in part by ONR/ARPA grants N0004-92-J-1512 and N00014-95-1-1088, and NSF grant IRI-9503312. And many thanks to Alex Rudnicky, Ronald Rosenfeld and Sunil Issar at CMU for providing the Sphinx-II system and related tools.

References

- Austin, J. L. 1962. *How to Do Things with Words*. New York: Oxford U. Press.
- James F. Allen. 1995. *Natural Language Understanding, Second Edition*, Benjamin-Cummings, Redwood City, California.
- James F. Allen, George Ferguson, Brad Miller, and Eric Ringger. 1995. Spoken dialogue and interactive planning. In *Proc. of the ARPA SLST Workshop*, Morgan Kaufmann, San Mateo, California.
- James F. Allen and C. Raymond Perrault. 1980. Analyzing intention in utterances, *Artificial Intelligence* 15(3):143-178
- S. Bennacef, L. Devillers, S. Rost, and L. Lamel. 1996. Dialog in the RAILTEL Telephone-Based System, *Proc. of the International Symposium on Dialogue, ISSD 6*, Philadelphia.
- Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer and Paul S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguis-*

- tics 16(2):79--85.
- Sandra Carberry. 1990. *Plan Recognition in Natural Language Dialogue*, MIT Press, Cambridge, Massachusetts.
- Herbert Clark, 1996. *Using Language*, Cambridge University Press.
- Cohen, P. R., and H. J. Levesque. 1990a. Intention is choice with commitment, *Artificial Intelligence* 42, 3.
- Cohen, P. R., and H. J. Levesque. 1990b. Rational interaction as the basis for communication In Cohen et al. (1990), 221--256.
- Cohen, P. R., J. Morgan, and M. Pollack. 1990. *Intentions in Communication*. Cambridge, MA: MIT Press.
- Phil R. Cohen and C.R. Perrault. 1979. Elements of a plan-based theory of speech acts, *Cognitive Science* 3:177-212
- George M. Ferguson, James F. Allen and Brad W. Miller, 1996. TRAINS-95: Towards a Mixed-Initiative Planning Assistant, in *Proc. Third Conference on Artificial Intelligent Planning Systems (AIPS-96)*.
- Richard Fikes and Nils Nilsson. 1971. STRIPS <<>>
- G. E. Forney, Jr. 1973. The Viterbi Algorithm. *Proc. of IEEE* 61:266--278.
- Grice, H. P. 1957. Meaning, *Philosophical Review* 66, 377--388. Reprinted in D. Steinburg and L. Jakobovits (eds.). 1971. *Semantics*. New York: Cambridge U. Press.
- Grice, H. P. 1975. Logic and conversation. In P. Cole and J. Morgan (eds.), *Syntax and Semantics*. Vol. 3: *Speech Acts*. New York: Academic Press, 41--58.
- Barbara Grosz and Candace Sidner. 1986. Attention, intention and the structure of discourse. *Computational Linguistics* 12(3).
- Barbara Grosz. 1974. The structure of task oriented dialog, *IEEE Symp. on Speech Recognition*. Reprinted in L. Polanyi (ed.). 1986. *The Structure of Discourse*. Norwood, NJ: Ablex.
- Barbara Grosz. 1977. The representation and use of focus in a system for understanding dialogs, *Proc. IJCAI*, 67--76.
- Barbara Grosz and Candace Sidner. 1990. Plans for discourse. In Cohen et al. (1990), 417--444.
- Barbara Grosz and Sarit Kraus. 1996. Collaborative Plans for Complex Group Action. In *Artificial Intelligence* 86(2), pp. 269-357
- Peter Heeman and James F. Allen. 1995. The TRAINS 93 Dialogues, TRAINS Technical Note 94-2, Department of Computer Science, University of Rochester. Speech data is available on CD-ROM from the Linguistics Data Consortium at the University of Pennsylvania.
- Peter Heeman and Graeme Hirst. 1995. Collaborating on Referring Expressions. In *Computational Linguistics* 21(3).
- Elizabeth Hinkelman and James F. Allen. 1989. Two Constraints on Speech Act Ambiguity, *Proc. ACL*
- X. D. Huang, F. Alleva, H. W. Hon, M. Y. Hwang, K. F. Lee, and R. Rosenfeld. 1993. The Sphinx-II Speech Recognition System: An Overview. *Computer, Speech and Language*
- Slava M. Katz 1987. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. In *IEEE Transactions on Acoustics, Speech, and Signal Processing* pp. 400-401.
- Diane Litman and James F. Allen. 1979. A plan recognition model for subdialogues in conversation. *Cognitive Science* 11(2):163-200
- Litman, D. J., and J. F. Allen. 1990. Discourse processing and commonsense plans. In Cohen et al. (1990), 365--388.
- Karen E. Lochbaum. 1994. *Using Collaborative Plans to Model the Intentional Structure of Discourse*. PhD Thesis, Dept. of Electrical Engineering and Computer Science, Harvard University PhD Thesis.
- Bruce Lowerre and Raj Reddy. 1986. The Harpy Speech Understanding System In *Trends in Speech Recognition* Speech Science Publications, Apple Valley, Minnesota. Reprinted in Waibel, 1990: 576-586.
- Perrault, C. R., and J. F. Allen. 1980. A plan-based analysis of indirect speech acts, *American Journal of Computational Linguistics* 6, 3--4:167--182.
- Eric K. Ringger and James F. Allen. 1996. A Fertility Channel Model for post-Correction of Continuous Speech recognition. in *Fourth International Conference on Spoken Language Processing (ICSLP '96)*. Philadelphia, PA.
- M.D. Sadek, A. Ferrieux, A. Cozannet, P. Bretier, F. Panaget, and J. Simonin. 1996. Effective Human-Computer Cooperative Spoken Dialogue: The AGS demonstrator, *Proc. of the International Symposium on Dialogue, ISSD* 6, Philadelphia.
- John Searle, 1969. *Speech Acts*, Cambridge University Press.
- John Searle, 1975. Indirect speech acts. In P. Cole and J. Morgan (eds.), *Syntax and Semantics*. Vol. 3: *Speech Acts*.

- New York: Academic Press, 59–82.
- Stefanie Seneff, Victor Zue, Joseph Polifroni, Christine Pao, Lee Hethington, David Goddeau, and James Glass. 1995. The Preliminary Development of a Displayless PEGASUS System. Proc. of the Spoken Language Systems Technology workshop, Jan. 1995. Morgan Kaufmann, San Mateo, California.
- Ronnie Smith and Richard D. Hipp, 1994. *Spoken Natural Language Dialog Systems: A Practical Approach*, Oxford University Press.
- Ronnie Smith, R. Hipp, and A. Biermann. 1995. An Architecture for Voice Dialogue Systems Based on Prolog-style Theorem proving, *Computational Linguistics*, 21(3).
- H. Teng, S. Busayapongchai, J. Glass, D. Goddeau, L. Hetherington, E. Hurley, C. Pao, J. Polifroni, S. Seneff and V. Zue. 1996. WHEELS: A Conversational System in the Automobile Classifieds Domain, Proc. of the International Symposium on Dialogue, ISSD 6, Philadelphia.
- Alex Waibel and Kai-Fu Lee, editors. 1990. *Readings in Speech Recognition*. Morgan Kaufmann, San Mateo, California.
- Daniel Weld. 1994. An Introduction to Least Commitment Planning, *AI Magazine*, Winter.

Appendix A1: The Complete Dialogue Showing Post-Processing

Here is a complete transcript of post-correction repairs for the sample dialogue.

REF: OKAY LET'S TAKE THE TRAIN FROM DETROIT TO WASHINGTON
SR: OKAY LET'S SEE CONTAIN FROM DETROIT TO WASHINGTON
PP: OKAY LET'S SEND CONTAIN FROM DETROIT TO WASHINGTON
REF: LET'S GO VIA TOLEDO AND PITTSBURGH
SR: LET'S GO VIA TOLEDO AND PITTSBURGH
PP: LET'S GO VIA TOLEDO AND PITTSBURGH
REF: NO LET'S TAKE THE TRAIN FROM DETROIT TO WASHINGTON VIA CINCINNATI
SR: NOW LET'S TAKE THE TRAIN FROM DETROIT TO WASHINGTON D S_X ANY
PP: NOW LET'S TAKE THE TRAIN FROM DETROIT TO WASHINGTON DO S_X ALBANY)
REF: OKAY THAT- THAT'S OKAY NOW
SR: OKAY IT THAT'S OKAY NOW
PP: OKAY IT THAT'S OKAY NOW
REF: OKAY NOW LET'S TAKE THE TRAIN FROM MONTREAL TO LEXINGTON
SR: UP OKAY NOW LET'S TAKE THE TRAIN FROM MONTREAL TO LEXINGTON
PP: UP OKAY NOW LET'S TAKE THE TRAIN FROM MONTREAL TO LEXINGTON
REF: LET'S GO VIA DETROIT
SR: LET'S GO P_M TO TRY
PP: LET'S GO P_M TO DETROIT
REF: YES NOW LET'S GO TO LEXINGTON
SR: YES NOW LET'S GO TO LEXINGTON
PP: YES NOW LET'S GO TO LEXINGTON
REF: OKAY NOW LET'S TAKE THE LAST TRAIN AND GO FROM ALBANY TO MILWAUKEE
SR: OKAY NOW ADD STATE THE LAST TRAIN AND GO FROM ALBANY TO O_O'S
PP: OKAY NOW I TAKE THE LAST TRAIN IN GO FROM ALBANY TO IS
REF: THE ENGINE AT ALBANY NEEDS TO GO TO MILWAUKEE
SR: ME A JET ADD ALBANY NEEDS TO GO TO MILWAUKEE
PP: ME THE JET AT ALBANY INSTEAD TO GO TO MILWAUKEE
REF: UH GO VIA BUFFALO
SR: I'D GO VIA BUFFALO
PP: UH GO VIA BUFFALO
REF: GO VIA SYRACUSE AND BUFFALO
SR: GO B_X SYRACUSE AT BUFFALO
PP: GO VIA SYRACUSE VIA BUFFALO
REF: THAT'S GOOD I'M DONE
SR: THAT'S COULD I CAN
PP: THAT'S GREAT UH CAN
REF: I'M DONE
SR: I_NEED DONE
PP: I'M DONE

Appendix A2: A Sample Keyboard Dialogue

U: Let's take the train in Charlotte to Lexington
S: *Yeah*
U: Can we go through Atlanta instead?
S: *Yes*
U: much better
S: *Oh?*
U: Let's move from Toronto to Washington
S: *What route would you like to get from Toronto to Washington*
U: through Montreal.
S: *Yes*
Is this ok?
U: good, now can we go down through Albany
S: *Huh?*
U: from Montreal, let's take the train south through Burlington and Albany.
S: *OK*
Is this ok?
U: Yes, now let's move the train in Boston to Washington.
S: *OK*
City New York is congested due to unusually heavy traffic.
An additional 5 hours will be needed to travel through there.
U: let's go around New York instead.
S: *OK*
U: How about going through Albany.
S: *Yep*
U: I'm sorry, can we move the train in Boston to Philadelphia instead.
S: *It would take a far more powerful and expensive computer than me to figure the answer to that one out.*
U: I'm sorry, can we move the train from Boston to Philadelphia.
S: *OK*
S: *But, Your routes cross at City Baltimore.*
Trains will take an additional 5 hours to move through the crossed cities.
U: Let's redo the route for Boston.
S: *What was that?*
U: Let's clear Boston's current route.
S: *Yeah*
U: Let's clear Boston's current route.
S: *Yes*
U: Let's move from Boston to Philadelphia
S: *OK*
U: I think we are done.

Appendix B: An Architecture for a Generic Dialogue Shell

James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson,
Lucian Galescu, and Amanda Stent
Dept. of Computer Science
University of Rochester
Rochester, NY 14627
James@cs.rochester.edu

B.1. Introduction

Different researchers use the term "dialogue system" in different ways. Beyond the belief that a dialogue system interacts with a human, there seems little commonality. For some, a dialogue system involves some mechanism to constrain the interaction, such as a script that specifies system prompts and user responses. The most common example of this type of system in use are the telephone interfaces based on keyed or spoken menu selection (e.g., "if you want your account balance, press or say 1, ..."). The goal here is to *restrict* the user's options in the interaction to simplify the language processing. For others, dialogue work aims to produce machines that can mimic human conversation. Work in this area aims to provide intuitive access to a wide range of applications, either over the telephone using voice only, or with multi-modal interaction at a computer workstation. The goal of this approach is to *expand* the user's options in the interaction.

We have the latter goal. We believe that such systems could be both feasible and cost-effective within the next decade. While speech-driven "menu-style" dialogue systems are growing in use today, they have their limitations. Unless the user needs to perform the most common and expected tasks, and can adapt to the preset method of performing those tasks, such interfaces can be a source of frustration. In addition, it appears that more complex tasks will be impossible to perform using such limited dialogue interaction. To support effective dialogue, the input language must be sufficiently general to express a wide range of different goals and courses of action. Effective interaction also requires an ability to support "mixed-initiative" interaction. The user must be able to "lead" the conversation in ways that best accomplish their goals, and the system should be able to take the initiative to speed up the solution when opportunities arise. Finally, in all but the simplest tasks, the system must be concerned with grounding: it must efficiently signal that the user was understood, and it must recognize when the user signals understanding or non-understanding.

But we are faced with a problem. Allowing unrestricted natural language dialogue would appear to require full human conversational competence, which does not seem feasible in the foreseeable future. We believe this argument is flawed, and that the required levels of conversational competence can be achieved in applications in the foreseeable future and at reasonable cost. The reason is that applications of human-computer interaction all involve dialogue focussed on accomplishing some specific task. We believe that the goal-seeking nature of such conversations naturally creates a specific genre of conversation, which we call *practical dialogues*. A practical dialogue

could involve tasks such as performing a simple transaction (e.g., ordering some merchandise), information-seeking (e.g., determining the arrival of flights, accessing medical information), engaging in problem solving (e.g., designing a kitchen), command and control (e.g., managing the response to a natural disaster), or tutoring (e.g., teaching basic concepts of mathematics). Our optimism depends on two hypotheses. The first concerns the complexity of practical dialogue:

The Practical Dialogue Hypothesis: The conversational competence required for practical dialogues, while still complex, is significantly simpler to achieve than general human conversational competence.

Even though a practical dialogue system might be possible to construct, however, it might still be too *expensive* to construct in practice. Again, based on our experience of building experimental systems in a number of domains, we believe that it will not. This suggests our second hypothesis:

The Domain-independence Hypothesis: Within the genre of practical dialogue, the bulk of the complexity in the language interpretation and dialogue management is independent of the task being performed.

If these hypotheses are true, then it should be possible to build a generic dialogue shell for practical dialogue. By “dialogue shell” we mean the full range of components required in a dialogue system, including speech recognition, language processing, dialogue management and response planning, built in such a way as to be readily adapted to new applications by specifying the domain and task models. This paper documents our progress and what we have learned so far based on building and adapting systems in a series of different problem solving domains. The first system was built in 1995, and the task was to find efficient routes for trains (the TRAINS domain). In 1996 and 1997, we evaluated the TRAINS system (Allen et al, 1996, Stent & Allen, 1997). The users had to find efficient routes for a group of trains (typically 3 trains) and avoid problem areas as they were discovered (e.g., congestion, tracks out). In a controlled experiment involving 80 sessions with 16 users who received less than three minutes of training, over 90% of the dialogue sessions resulted in successful plans without any intervention at all from the experimenter.

Based on this experience, we designed a new architecture to better support handling different domains. This resulted in TRIPS (The Rochester Interactive Planning System). TRIPS is designed to support plan-based tasks, and we have built versions of the system in several domains. TRIPS-PACIFICA involves evacuating people off an island in the face of an impending hurricane and is robust for naïve users. TRIPS-CpoF was a limited scripted-only demonstration system that involved planning the deployment of troops. TRIPS-AMC involves planning airlifts and investigates how “third-party” back-end systems could be integrated into the TRIPS architecture. The Monroe domain, under current development, involves coordinating emergency vehicles in response to simulated 911 calls and serves as our current experimental domain. This is an exercise in scalability as the task involves building and evaluating a robust system in a significantly larger domain than we have previously worked on. Table 1 summarizes the domains and the status of each project.

Domain	Date	Task	Goal	Status
TRAINS	1995-7	Finding efficient routes for trains	Robust performance on a very simple task	Robust performance (>90% success rate)
PACIFICA	1997-8	Evacuating people from an island	Robust performance in a task requiring explicit planning	Demonstration system supports untrained users
CpoF	1998	Deployment of troops in a military situation	Scripted demonstration in a military relevant task	Scripted interaction only
Monroe	1999-	Coordinating responses to emergencies in Monroe County	Robust performance on a dynamic, mixed-initiative task involving planning, monitoring and replanning, larger domain	In development for robustness evaluation
AMC	1999-	Planning airlifts using an airlift planning system	Demonstrate capability to use "third-party" planning systems	Initial demonstration completed, work on extensions continuing
Kitchen	planned	Planning Kitchen Design	Robust performance on a significantly different task	Planned for development

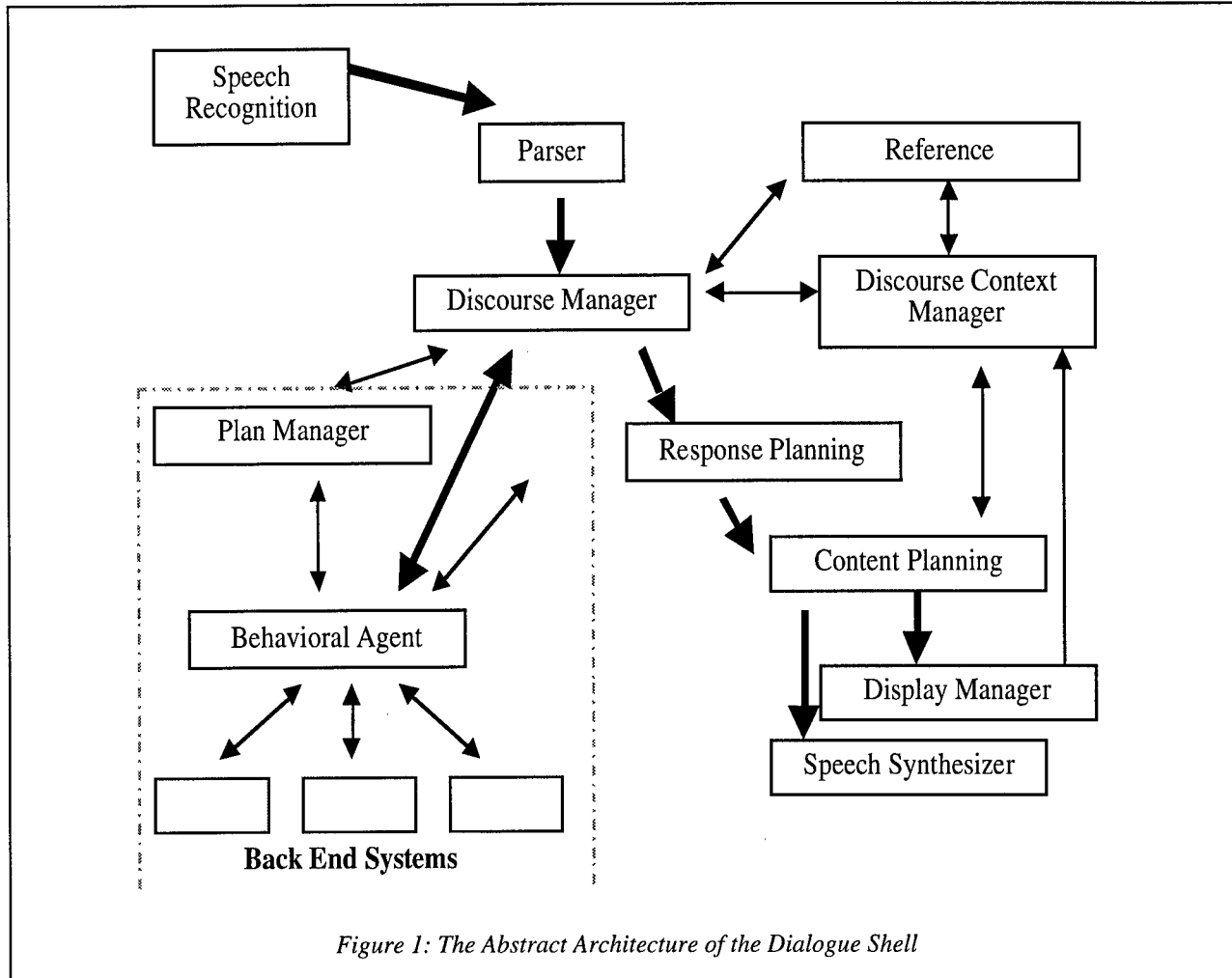
B.2. A Generic Dialogue Shell

In designing our architecture, we have kept the following challenges in mind:

- *Intuitive Natural Input*: Whether using the keyboard or speech, a human must be able to specify their needs simply and directly. Unconstrained natural language is one of the few viable options, especially for telephone-based interfaces.
- *Robustness*: The system must continue the dialogue coherently despite speech recognition errors and misunderstandings, and resolve such problems within the dialogue itself.
- *Mixed-Initiative Interaction*: The system must support mixed-initiative interaction, in which either the system or user may take "control" of the dialogue at different times and have freedom to direct the conversation towards best achieving their goals.
- *Intention Recognition*: The system must be able to identify the user's intention, namely, to recognize what the user wants to do. In domains with fairly limited tasks, this may be simple to do. In more complex domains, the problem becomes quite challenging.
- *Effective Grounding*: The system must be able to maintain a sense of mutual understanding using methods that are natural to human conversation. It must be able to clarify situations and correct misunderstandings, and recognize when the user does so.

- *Topic Tracking*: In the simplest domains, there may be only one task and the topic of conversation remains essentially fixed. In all other cases, the system will need to identify topic flow during the conversation.
- *Dialogue-based Response Planning*: The system must be able to provide appropriate levels of information in its responses, possibly in an incremental fashion in an extended sequence of interactions.
- *Portability*: Dialogue system components should either be usable in any domain as is, or be easily adaptable to work in new domains.

In the remainder of this paper we will address architectural concerns in designing and building a generic dialogue shell. We will first describe the component-level architecture, which reflects our experience in building systems, with a special focus on the need to separate domain-independent aspects of the system from the domain-specific components that create a specific application domain. We then describe our programming level architecture, which has proven very effective in supporting system development and porting to new applications. Finally, we will describe a few specific components in the system to illustrate our approach of developing domain-independent components that can then be rapidly tailored to a specific domain.



B.3. A Generic Architecture for Dialogue Systems

Simple dialogue systems may consist of a fixed sequence of processing stages, starting with speech recognition, then parsing/analysis, dialogue management and response generation. As the dialogues to be handled become more complex, however, such a simple architecture does not work effectively. For instance, our current system shell under development has six separate modules that together provide the dialogue management: discourse context management, reference resolution, intention recognition, the behavioral agent, the plan manager, and response planning. Each of these plays a distinct role, and while they could be collapsed together for a particular application (as we did in the original TRAINS system), such monolithic modules are hard to construct and debug, and are difficult to modify for a new task and domain. Figure 1 shows the core set of modules in the generic dialogue shell, and Table 2 gives a brief description of each. The heavier arrows in Figure 1 show the main flow of processing from an input utterance to a response, and the lighter arrows indicate the main interactions along the way. In addition, all modules have access to a common se-

Module	Function
Speech Recognition (SR)	Transforming speech input into a word stream or word lattice
Parser	Transforming the SR output into interpretations, each a set of conventional speech acts, using full and robust parsing techniques
Reference Manager (REF)	Identifying the most salient referents for referring expressions such as noun phrases
Discourse Context Manager	Maintaining the global (topic flow) and local (salience with a topic) discourse context
Discourse Manager (DM)	Identifying the intended speech act, current task, current step in the current task, and system obligations arising from the dialogue
Behavioral Agent (BA)	Determines system actions (e.g., answer a question, notify of a problem, request clarification); Manages the interface to the back-end systems.
Plan Manager	Constructing, modifying, evaluating, and executing plans (whether they are the subject of the conversation or the task being executed)
World KB	Maintains a description of the current state of the world under differing assumptions (e.g., based on different plans or hypotheses)
Response Planner	Determining the best communicative act(s) (and their content) to accomplish the system's current goals and discourse obligations
Content Planner	Determining how to realize the planned speech acts
Display Manager	Managing the visual presentations given the available hardware.

Table 2: The Key Modules in the Dialogue Shell in the Abstract Architecture

mantic hierarchy and to a world KB manager that handles queries about the current situation, managing the interfaces to domain dependent reasoners and knowledge bases as needed.

One of the key things to note about this architecture is the separation of the basic dialogue system components from the more domain-specific components that provide the application (shown within the dotted lines at the lower left corner of Figure 1). To illustrate this separation, consider a specific example: a travel-agent application. The back-end would provide schedule and reservation information, booking, and so on, much as current computer systems provide to human travel agents. The behavioral agent and plan manager would be driven from a specification of desired behavior of the system as a travel agent, including the actions it typically will be asked to perform (e.g., what information is relevant to the customer when planning a trip), what obligations it has (e.g., find the customer the most convenient flights, say cheapest, or fastest), and a specification of how to perform actions (e.g., to book a ticket, first get credit card information, then interact with the reservation system, confirm booking with user, etc). As another example, in the TRIPS-PACIFICA system, the back-end systems included a movement planner (choosing movement actions including the vehicles and cargoes involved), route planning (based on map and vehicle capabilities), and scheduling (based on nominal travel times). The Behavioral agent knew the various operations a person might request (e.g., introduce a new subgoal, modify an existing plan, evaluate a plan, e.g., how

long will that take?), as well as how to perform actions (e.g., to develop a plan, first determine the movement actions required, then call router to find routes, and then call the scheduler to produce a time-line), and then convey this information back to the user.

Separating the Behavioral Agent from the dialogue management is also key to supporting mixed-initiative interaction. The behavioral agent may have goals independent of the current conversation that might influence its response. For example, in an emergency management task, the behavioral agent may decide that it is more important to notify the user that an ambulance has become inoperative rather than to answer the user's current query about the weather forecast. Thus it might ignore the question temporarily (still retaining the obligation to answer later). Determining when to do this, of course, will be a domain-specific decision.

The Behavioral Agent also serves to encapsulate the domain-specific information, for the rest of the system interacts with the back-end systems only via the behavioral agent. By defining a generic interface to the behavioral agent in terms of plans, action, goals, and so on, most of the system can be built independent of any specific domain. For example, The Discourse Manager (DM) coordinates a range of processes to recognize the user's intentions underlying the utterance and to compute new discourse obligations (e.g., if asked a question, one should respond (cf. Traum & Allen, 1994)). The DM first receives input in the form of surface speech acts that are computed by the parser using the output from the speech recognition system. It is driven by a set of interpretation rules that match the surface speech acts and invoke modules such as the Reference Manager, Context Manager, Plan Manager and the Behavioral Agent to produce and/or evaluate possible interpretations. It might seem that the DM needs domain-specific information to perform its task. Instead, however, it deals with abstract intentions such as introducing a new goal, modifying an existing plan, and requesting background information. Intentions of this form are evaluated by the behavioral agent and plan manager with respect to the specific domain and the results are passed back to the DM. The DM does not have to know the details of the specific domain.

B.4. Program Level Infrastructure

We now turn to the second level of architecture, namely the programming infrastructure used to build dialogue systems. Portability to new domains and flexibility in accommodating new components requires a clear and explicit separation of responsibilities between the components. Not coincidentally, this is also good software engineering practice. Importantly for our team of researchers, it also allows work on individual components or small groups of components to proceed independently, with later integration into the complete system (possibly even at run-time).

Our program-level architecture consists of a set of loosely-coupled, heterogenous components that communicate by exchanging messages. Over the past several years, we have developed an extensive robust infrastructure for deploying components.

Why a message-passing architecture?

Before describing the infrastructure, it is worth considering why we chose an explicit message-passing framework for inter-component communication rather than an alternative such as remote procedure calls or object-oriented method calls.

The first reason is that explicit message-passing provides platform- and language-independence. In our work, we use a variety of languages and platforms including Lisp, C, C++, Java, and Perl on Unix, Windows, and Macintosh platforms. TRIPS components can also communicate with other systems via agent frameworks such as the Open Agent Architecture (OAA) (Cohen et al, 1994) and object frameworks such as CORBA. From the outset, we didn't want to restrict ourselves to a particular object framework or communication model.

The second reason for using explicit message-passing is the separation of transport and content information. The transport mechanisms and protocols are supported (and enforced) by the infrastructure, while the content is negotiated between the components. This negotiation can be at compile-time, in the case of objects, or at run-time in the case of agents. But the fact that message-passing separates transport and content decisions allows us to build a robust communication infrastructure without requiring prior universal agreement about content.

Third, and extremely important for us as researchers and system developers interested in rapid portability, is that explicit message-passing provides easy access to information needed for debugging. The message traffic in a run of the system is logged and can be used for post-mortem debugging and even replaying entire sessions.

Syntax and Semantics of Messages

In the TRIPS message-passing framework, components exchange messages using either the Knowledge Query and Manipulation Language (KQML) (Labrou and Finin, 1997), designed as part of the DARPA Knowledge Sharing Effort, or the Agent Communication Language specified by the Foundation for Intelligent Physical Agents (FIPA ACL, 1999). These languages are quite similar and both use a Lisp-like s-expression syntax and are based loosely on the semantics of speech acts (e.g., Searle, 1969).

A message in these languages consists of a verb (or performative), indicating the speech act intended by the message, followed by a set of parameters specifying particulars of the message. Examples of KQML performatives include "TELL" and "ASK-IF". Messages can be addressed to other agents, and there is support in the specifications for connecting individual messages into conversational threads.

Importantly, neither KQML nor FIPA ACL specifies the semantics of the content of the messages. That is, there is a parameter with which the content of the message can be specified, but this parameter is not interpreted by the message-passing infrastructure. While this makes it difficult to specify and verify agents in terms of their communicative behaviors, it does make it possible to use these frameworks without solving a host of difficult problems in reasoning about other agents (cf. Cohen and Levesque, 1990). The fact that components developed using KQML or FIPA ACL share

a syntax and a basic agreement about the meaning of the performatives generally makes their integration easier than if they communicated only via specialized interfaces.

The TRIPS Facilitator

The TRIPS message-passing infrastructure is based on a hub topology. The central node in the network of components is the TRIPS facilitator. The choice of a hub topology was a pragmatic one. While such an organization does lead to a possible single point of failure for the network if the hub goes down, in practice the advantages (such as support for validation and logging) are more important for a research system.

The main job of the Facilitator is to route each message to its intended receiver. In so doing, it performs full syntactic validation of the message with respect to the underlying protocol (e.g., KQML). This again simplifies component development by removing much of the burden of protocol-level error-checking. To perform the routing, the TRIPS Facilitator allows clients to register one or more names. The Facilitator also allows the online specification of "client groups" to which individual clients can subscribe. In the TRIPS prototype system, these groups are used to describe categories of services, such as "user input" or "display". The Facilitator accepts group names (i.e., service classes in the TRIPS prototype) as valid recipients for messages, and routes such messages to all members of the group.

These capabilities support an extended form of point-to-point addressing between components exchanging messages. However, in a system like TRIPS where the components are effectively members of an agent society, broadcasting is a common way of informing other agents of new information or requesting services. The TRIPS Facilitator therefore supports two separate forms of broadcast: 1) a "true" broadcast, which is used for various control messages, such as to indicate that a conversation is starting or has ended, and 2) a "selective broadcast", a subsumption mechanism in which components can indicate interest in the output of a particular client or client group. Combined with the use of client groups, selective broadcast provides a powerful and flexible communication framework (for example, the Parser can indicate interest in the "user input" group, of which the Speech Recognizer is a member, along with other clients).

In summary, the TRIPS Facilitator provides robust message-passing facilities built on TCP sockets. It provides naming services, content- and service-based addressing, broadcast, selective broadcast, and logging. It has many similarities with agent-based communication languages like OAA and can easily be extended to interact with OAA agents. We differ significantly from the Communicator Architecture (Goldschen & Loehr) in that we do not have any concept of programming the Hub. We believe that the communication infrastructure should be general and all control issues should be handled within the modules.

B.5. Issues in Porting to New Applications

The final issue we will discuss is the issue of adapting the shell to new domains. As discussed above, our general strategy is to develop generic components that are domain-independent yet limited to practical dialogues. In some cases, a generic component can work “as is” in domain. In others, we need techniques for rapidly specializing the components to perform effectively in each specific domain. We will discuss a few specific examples here as there is not the space to cover every aspect of the system.

Speech Recognition

We use a general-purpose speech recognizer, Sphinx-II (Huang1993) and generic acoustic and pronunciation models to achieve generality. We then tailor the lexicon and the language model to the task domain in order to achieve good recognition performance. This is especially important because our dialogues involve more spontaneous speech than applications such as dictation.

Most approaches to building language models for new domains are framed as adaptation problems: assume the availability of a good general model that can be made more specific by incorporating knowledge from a text corpus in a new domain. However, these approaches require the existence of a corpus for the new domain, which may become prohibitively lengthy and expensive. We have successfully built language models for new domains in a very short time by applying two techniques (Galescu et al, 1998). First, in the absence of text data in the task domain, we generate an artificial corpus from a hand-coded context-free grammar (CFG) that is easily adaptable to new domains, and train a language model (LM) on this corpus. Second, we use a class-based approach to LM adaptation from out-of-domain corpora that allows us to re-use the dialogue data collected from other practical dialogue domains to build models for new domains.

The first technique involves generating an artificial corpus by Monte Carlo sampling from a hand-coded task-specific context-free grammar. The purpose of the artificial corpus is to provide a source of plausible word collocations for the new domain. As such, a simple grammar based on a blend of syntactic and semantic categories is sufficient, and has the advantage of being very easy to write. Our grammars contain just a few hundred rules. We built the first CFG (for the Pacifica domain) starting from a few sentences (e.g., from a script) in a manner similar to Rayner (1997).

The second technique we use to obtain training material for language modeling is based on reusing data from other domains. Language models trained on the data from other domains might actually give worse performance than not using any language model at all. Therefore, we need to provide a transformation between language models in the remote domain and language models in the target domain. To do this we use a class-based approach. The general procedure for generating class-based n-gram models (Issar 1996) follows three steps: 1) Tagging the corpus according to some predefined word-class mapping; 2) Computing a back-off n-gram class model from the tagged text corpus; and 3) Converting back to a word model using the word-class mappings in the class tag dictionary. We achieve the adaptation by using a tag dictionary specific to the remote

domain for tagging the corpus (in step 1), and using a tag dictionary specific to the target domain to obtain a word-based LM from the class-based LM (in step 3). The success of this technique depends heavily on the compatibility between the two tag dictionaries being used. To achieve this kind of compatibility, we maintain a domain-independent tag dictionary (containing function words, pronouns, many common-use words and phrases of basic conversational English), and a specialized tag dictionary for each task domain. Thus, whenever we work on a new domain, we only need to adapt the specialized tag dictionary.

One disadvantage of this approach is that no statistics are available for words in the target domain associated with tags that don't appear in the remote corpus. We alleviate this problem by interpolating language models from several practical dialogue domain.

We have used the above techniques for building initial language models in all domains to which we have recently ported the TRIPS system (see Table 1). The most thorough evaluation was done on the Pacifica domain. We summarize in Table 3 the main results. Full details can be found in (Galescu et al, 1998). Listed are perplexity (PP) and word error rate (WER) figures for various models¹. 1) NULL - no model is used. This would be the default when no information about the new domain is available; 2) artificial corpus - the initial model built from an artificial corpus; 3) adapted OOD data - the initial model built using the class-based adaptation from out-of-domain (OOD) corpora. The corpora used were: the ATIS corpus (Dahl et al, 1992), the TRAINS95 transportation scheduling domain corpus (Allen et al, 1996), and the TDC human-human spoken dialogue corpus (Heeman & Allen, 1995); 4) combined - an initial model that combines by linear interpolation the two other initial models. The interpolation weights were chosen to optimize PP on the transcriptions of the first few conversations with the system. Choosing equal weights provided close performance figures; 5) Pacifica - a model built from transcriptions of actual conversations with a fully operational system in which the language model was the combined model.

Model	PP	WER
NULL	1862.0	56.9
Artificial corpus	57.94	28.2
Adapted OOD data	92.01	33.7
Combined	32.86	26.3
Pacifica	15.92	18.8

Table 3: Performance of the various initial models compared to the NULL and Pacifica models.

As can be seen from Table 3, the techniques described above for building initial models for new domains provide reasonable performance. After deploying the system, as text from the target domain becomes available, it can be used for building a domain-specific model. This may be used for further adaptation of the initial model (Rudnicky, 1995), or may replace the initial model.

¹ All models are open vocabulary bigram back-off models, with Witten-Bell discounting (Witten & Bell, 1991) and built with the CMU Statistical Language Model Toolkit (Clarkson & Rosenfeld, 1997) and tools of our own.

Parsing

Working with real-time dialogue places many demands on the parser. If an analysis can be found, it must be done quickly so that the user does not have to wait too long. At the same time, we want the grammar to cover a wide range of grammatical constructs, but extending the grammatical coverage often decreases the efficiency of the parser. We address this problem by extensive use of selectional (i.e., semantic) restrictions. While it has been argued that selectional restrictions are problematic as a general theory of semantics, we have found them to be practical and effective for specific domains. The challenge is to keep a general domain-independent grammar and lexicon for portability and then adapt it to a new domain for accuracy and efficiency. The parser uses a chart-based best-first algorithm that accepts input incrementally. The grammar is a fairly standard feature-based context-free formalism. The parser and grammatical formalism are based on Allen (1995).

We start with a grammar and a core lexicon that describes general English usage in practical dialogues and then obtain a domain-specific parser by compiling in a domain semantics. For example, there are general patterns for the syntax of a verb phrase: a verb can only take a certain number of complements that are well-defined constituents. Princeton WordNet (Miller, 1995) identifies 35 verb frames encountered in English. While some domains may not involve all these frames, it is reasonable to assume that the number and syntactic types of verb arguments is domain independent. Therefore, they are part of the core lexicon and grammar. Similarly, there are semantic properties that stay the same in all domains. A car is a physical object in all domains, and it cannot conceivably be a living entity.

Specific domains differ primarily in the semantics they assign to lexical items. Not only do words have different meanings in different domains, but the notion of a relevant semantic property varies. For example, in the Pacifica domain it is important to distinguish fixed objects such as cities from movable objects such as people and cargo. On the other hand, in a kitchen design system we will need to distinguish between furniture and appliances, something that is not at all important in Pacifica, where they will be all treated as cargo. In a similar vein, the verb "move" in Pacifica refers to a transportation action, whereas in the Kitchen design domain it refers to a change in the kitchen plan, and transportation actions are unknown. In Pacifica, a window in a house would be fixed (not movable), whereas in the Kitchen domain it is movable (in the design plan).

To build a domain specific grammar, we define mappings from the domain-independent representations to domain-specific predicates. For example, the generic grammar contains a domain-independent verb class "MOVE". In the Pacifica domain, transportation actions are important and defined by a domain-specific predicate TRANSPORT. As a result, we produce domain-specific semantic restrictions on the roles for MOVE-class verbs: e.g., the *Instrument* is required to be a vehicle, the *Theme* is restricted to be movable (also a domain-dependent feature that selects the set of objects that can be moved in the domain). We have now significantly reduced the number of sentences that could be parsed, which increases the efficiency of the parser, improves structural disambiguation, and provides strong semantic guidance for robust processing to recover from speech recognition errors. In the kitchen domain, on the other hand, MOVE would map to a kitchen plan

modification action, where the *agent* can only be one of the dialogue participants, the *Theme* is a "movable" design element in the design plan, while the *instrument* will be always absent. The grammars that result from those two specializations result in very different sentences being understandable, giving us the advantage of domain-specific semantic grammars without the complexity of having to define different grammar rules by hand.

Reference Resolution

Resolution of referring expressions is part of the semantic interpretation phase of the DM. In our implementation, referring expressions are resolved by first constructing a list of known properties of the referent, calling knowledge managers within the system (such as the context manager and display manager) to return entities matching those properties, then calculating a confidence rating that each matching entity is the correct referent. By depending on other knowledge managers for any domain-specific information, the reference resolution module can remain effectively domain-independent in generating possible candidates.

Calculation of the confidence rating varies depending on the form of the referring expression (RE) used (e.g., pronoun, definite description, indefinite) and is domain independent. Returning a list of possible referents, rather than just the one most probable referent, gives the DM more flexibility in combining the results with information from other sources.

Resolution of referring expressions can take advantage of semantic information if it is available in the system. For example, verb semantic restrictions computed by the parser can be used to limit the possible semantic type of allowable referents. To do this requires access to a semantic hierarchy that is shared by all modules in the system.

In practical dialogues, we find a larger variety of referring expressions and referring behavior than one finds in other language genres. Our team at the University of Rochester analyzed two spoken corpora for their referential behavior (Byron, 1999a). While 89% of the pronouns in our spoken monologue corpus were co-referential with a noun phrase, only 25% of the pronouns in our practical dialogue corpus were. That means that using a pronoun resolution technique that relies solely on identifying noun phrase antecedents in the prior discourse (a commonly used technique in language understanding systems) is entirely inadequate for practical dialogs. In practical dialogues, reference to events, propositions, entire stretches of discourse, etc. is much more common, therefore the pronoun resolution technique must allow a wide variety of candidate referents. Practical dialogues also contain much more demonstrative reference. In the TRAINS corpus (Heeman & Allen, 1995), 50% of the pronouns are demonstratives, compared to less than 10% in the spoken monologue corpus. This is important because demonstrative pronouns can refer to a wider range of entities than can definite pronouns (Byron & Allen, 1998), and resolution of demonstrative pronouns is very rarely addressed in the literature. A much more sophisticated model, one that computes a semantic interpretation of the input text, is needed to resolve demonstrative pronouns (cf. Byron, 1999, Eckert & Strube 1999).

Content planning and generation

In the current TRIPS system, the response planner receives conversational goals from the discourse manager (which is driven by directions from the behavioral agent). It selects content for output and passes the generator a set of role-based logical forms with associated speech acts. The generator decides which logical forms to produce and how to sequence them, and then passes commands on to modality-specific generators (prosodically-annotated textual sentences are sent to the speech synthesizer; commands to display maps and charts are sent to the Display Manager).

Our current language generator uses a domain-specific template-based approach. Templates work well in near-fixed-initiative situations where there are few types of system utterances; they are fast and flexible. Grammar-based approaches work well when broad language coverage is needed, but are not usually fast enough for use in real-time dialog systems.

Our goal in the next version of TRIPS is to develop a conversational agent capable of generating varied and complex content in a mixed-initiative fashion. Accordingly, our proposed generation framework differs significantly from our existing one. It is based on theoretical and empirical understandings of what dialog contributions involve (Traum & Hinkelman, 1991) and is designed to support incremental, real-time generation; broad language coverage; and flexibility in modifying system behavior and switching to new domains. We consider the three stages of generation (planning intentions, planning content and planning form) as three different aspects of utterances that have to be planned, possibly simultaneously (Reithinger, 1990, Desmedt et al, 1993). Grounding and turn-taking acts, for instance, involve the planning of intentions only. Also, these acts often begin a turn, so generating these acts quickly can give a conversational agent time to produce other acts that may involve more processing. Because the form of such acts can be selected from a set of conventional forms, a template-based approach can both be domain-independent and be fast enough for real-time interaction.

Those utterances that speakers produce to fulfill intentions arising directly from the domain or the task being solved often have content that must be expressed. The form may or may not be important. We will generate these types of utterances using templates when the forms are limited, but use grammar-based generation in the more complex cases. Other utterances are produced primarily to complete an argumentation act. Their production involves the planning of intentions, semantic content and surface form, and they would usually need to be generated using a grammar.

Our proposed generator architecture is described in detail in Stent 1999a. The new response planner will plan the system's intentions for the continuing dialogue by considering a set of possible interpretations from the DM and the Behavioral Agent. It must ensure that none of the selected goals conflict or are redundant. Generation goals are represented as sets of intention by content pairs. The generator maintains a list of these sets (we call this list the intention-set). Each set is sent to all the generation modules. The output from each module is a surface form and a set of intentions fulfilled by that form. A gate-keeper at the end removes intentions from the intention-set as they are fulfilled. It can also add sets of intentions to the intention-set, for instance to keep the turn or if the

agent is interrupted. Finally, it can minimize over-generation by selecting which results to produce, if it gets simultaneous results that satisfy the same intentions.

Consider an example. Imagine a user has just made a statement to the agent. The agent wants to acknowledge part of the statement (grounding) and ask a question about another part. So the intention-set looks like: {take-turn, acknowledge(Utt1), info-request(Content)} (items with initial capital letters are variables). This set gets passed to all modules. The turn-taking module returns "Uh" for take-turn and the grounding module returns "Okay" for take-turn and acknowledge(Utt1). The gate-keeper therefore removes take-turn and acknowledge(Utt1) and produces "Okay". If a pause of more than, say, half-a-second ensues, the gate-keeper might add the intention keep-turn to the intention-set which will feed it to the various modules. However, happily the gate-keeper quickly receives a result for info-request(Content) which it produces, removing that intention (and therefore the whole set) from the intention-set.

Real-time generation is very important in the context of dialog. We believe our architecture will allow for real-time responses with grounding and turn-taking acts, "buying" the system more time to plan utterances that convey complex content.

Conclusions

The TRIPS project is very much a work in progress, and many of our ideas presented here still await a rigorous formal evaluation. The architecture described here was developed and refined in response to the problems we found when building end-to-end systems, and in porting the system to new domains. The TRIPS project shares many of the goals of the Communicator project (Seneff et al, 1998). In fact, we have demonstrated that we are able to meet many of these goals: we currently support four domains; the KQML communication language and agent architecture of our system allows us to use a plug-and-play approach to development and facilitates the incorporation of different modalities; and we have identified and developed key domain-independent components that can be specialized to new domains fairly easily.

References

- Bordegoni, M. et al. A Standard Reference Model for Intelligent Multimedia Presentation Systems, *Computer Standards and Interfaces*, 18 (6), 1997. 477-496
- De Smedt, K. and H. Horacek and M. Zock, "Architectures for Natural Language Generation: Problems and Perspectives", *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, Springer-Verlag, 1996.
- Byron, Donna K. and James F. Allen, "Applying Genetic Algorithms to Pronoun Resolution", *AAAI-99*, 1999.
- Byron, Donna K. and James F. Allen, "Resolving Demonstrative Pronouns in the TRAINS93 corpus", *daarc2*, 1998, 68-81.

- Byron, Donna K. "Analysis of pronominal reference in two spoken language collections: TRAINS93 spontaneous task-oriented dialogue and Boston University radio news prepared monologue", University of Rochester, Dept. of Computer Science Tech Report 703, October 1999a.
- Byron, Donna K. "Resolving Pronominal Reference to Abstract Entities: Thesis Proposal", University of Rochester, Dept. of Computer Science Tech Report 714, June 1999b.
- Clark, Herbert and Edward F. Schaeffer. Contributing to Discourse, *Cognitive Science* 13, 1989. 259-294.
- Cohen, P. R. , A. J. Cheyer, M. Wang, and S. C. Baeg, "An open agent architecture," in AAAI Spring Symposium, Software Agents, pp. 1--8, 1994.
- Clarkson, P. and R. Rosenfeld, Statistical Language Modeling Using the CMU-Cambridge Toolkit, *Proc. Eurospeech '97*, Greece, 1997. 2707-2710.
- Dahl, D.A. et al. Expanding the Scope of the ATIS Task: The ATIS-3 Corpus, *Proc. ARPA Human Language Technology Workshop '92*, 1992. 45-50.
- Eckert, Miriam and Michael Strube. "Resolving Discourse Deictic Anaphora in Dialogs", Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL '99), 1999.
- Galescu, L. Eric Ringger and James Allen. Rapid Language Development for New Task Domains, to appear in *Proc. ELRA First intl Conf. on Language resources and Evaluation*, Granada, Spain, May 1998.
- Goldschen, A. and Loehr, D. The Role of the DARPA Communicator Architecture as a Human Computer Interface for Distributed Simulations, document, MITRE Corporation.
- Hobbs, J. R. Resolving pronoun references, *Lingua* 44, 1978. 311-338. Reprinted in *Readings in Natural Language Processing*, B.Grosz and K.Sparck Jones and B.Webber (eds), Morgan Kaufmann, 1986. 339--352
- FIPA, Agent Communication Language, FIPA Spec 2 - 1999 (draft version 0.2), Foundation for Intelligent Physical Agents, Geneva, Switzerland, 16 April 1999.
- Heeman, P.A. and J. Allen. The TRAINS 93 Dialogues, TRAINS TN 94-2, University of Rochester, Computer Science Dept., 1997. Corpus available from LDC.
- Huang, X. and F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee and R. Rosenfeld. The SPHINX-II speech recognition system: an overview, *Computer Speech and Language*, 7 (2), 1993. 137-148
- Issar, S. Estimation of Language Models for New Spoken Language. *Proc. ICSLP'96*, Philadelphia, PA. 1996. 869-872.

- Labrou, Yannis and Tim Finin, *_A Proposal for a new KQML Specification_*, TR CS-97-03, February 1997, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250.
- Miller, G.A. WordNet}: A Lexical Database for English, *Communications of the ACM* 38 (11), 1995. 39-41
- Miller G.(ed). WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 1990.
- OMG, The Common Object Request Broker: Architecture and Specification (Revision 2.3), Object Management Group, June 1999.
- Rayner, M. and Carter, D. Hybrid Language Processing in the Spoken Language Translator. *Proc. ICASSP'97*. 1997. 107-110
- Reithinger, N. "POPEL -- A Parallel and Incremental Natural Language Generation System", in . L. Paris and Swartout, W. R. and Mann, W. C., *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, Kluwer Academic Publishers, 1991. 179-199.
- Rudnicky, A.I., Language modeling with limited domain data. *Proc. ARPA Spoken Language Technology Workshop*, 1995.
- Searle, J. R. 1979. "A taxonomy of speech acts." In J. Searle (ed.), *Expression and Meaning*. Cambridge, U.K.: Cambridge U. Press, 1-29.
- Seneff, S., E. Hurley, R.Lau, C.Pao, P. Schmid, and V. Zue. "Galaxy-II: A Reference Architecture for Conversational System Development", *Proc. ICSLP 98*, Sidney, Australia, 1998.
- Traum, D.R. and James Allen, "Discourse Obligations in Dialogue Processing", *Proc., 32nd Annual Meeting of the ACL*, 1-8, Las Cruces, NM, June 1994.
- Traum, D. R., and E. A. Hinkelman. "Conversation acts in task-oriented spoken dialogue," *Computational Intelligence* 8, 3. 1992.
- Vossen, P. EuroWordNet: a multilingual database for information retrieval. *Proceedings of the Delos workshop on Cross-language Information Retrieval*, 1997.
- Witten, I.T. and Bell, T.C. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory* 37 (4). 1085--1094.
- Zue, V. and Glass, J. and Goodine, D. and Leung, H. and Phillips, M. and Polifroni, J. and Seneff, S. Integration of Speech Recognition and Natural Language Processing in the MIT VOYAGER System. *Proc. ICASSP'91*. 1991. 713—716.

DISTRIBUTION LIST

addresses	number of copies
ATTN JOSEPH CAROLI AFRL/IFTB 525 BROOKS RD ROME NY 13441-4505	7
ATTN DR JAMES ALLEN DEPT OF COMPUTER SCIENCE UNIV OF ROCHESTER ROCHESTER NY 14627-0226	5
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 3725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	1
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P. STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/MLME 2977 P STREET, STE 6 WRIGHT-PATTERSON AFB OH 45433-7739	1

AFRL/HESC-TDC 1
2698 G STREET, BLDG 190
WRIGHT-PATTERSON AFB OH 45433-7604

ATTN: SMDC IM PL 1
US ARMY SPACE & MISSILE DEF CMD
P.O. BOX 1500
HUNTSVILLE AL 35807-3801

TECHNICAL LIBRARY D0274(PL-TS) 1
SPAWARSYSCEN
53560 HULL ST.
SAN DIEGO CA 92152-5001

COMMANDER, CODE 4TL000D 1
TECHNICAL LIBRARY, NAWC-WD
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100

CDR, US ARMY AVIATION & MISSILE CMD 2
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSAM-RD-OB-R, (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5000

REPORT LIBRARY 1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

AFIWC/MSY 1
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016

ATTN: KAROLA M. YOURISON 1
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE
PITTSBURGH PA 15213

USAF/AIR FORCE RESEARCH LABORATORY 1
AFRL/VSOSA(LIBRARY-BLDG 1103)
5 WRIGHT DRIVE
HANSCOM AFB MA 01731-3004

ATTN: EILEEN LADUKE/D460
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

1

OUSDP)/DTSA/DUTD
ATTN: PATRICK G. SULLIVAN, JR.
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

1

***MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)***

*The advancement and application of Information Systems Science
and Technology to meet Air Force unique requirements for
Information Dominance and its transition to aerospace systems to
meet Air Force needs.*